2-3-00

A

PATENT

APPLICATION FOR U.S. PATENT
TRANSMITTAL FORM

THE COMMISSIONER OF PATENTS
AND TRADEMARKS
Washington, D.C. 20231

Docket No. _____-CS

Sir:

Transmitted herewith for filing is the patent application of:

Inventor(s): NORTH, Gregory Allen, PERRY, Matthew Richard, and KIRCHER, Brian Christopher

For: CIRCUITS, SYSTEMS AND METHODS FOR INFORMATION PRIVATIZATION IN PERSONAL ELECTRONIC APPLIANCES

Enclosed are: _31_ Sheets of Drawings

An assignment of the invention to: Cirrus Logic, Inc.

LARGE ENTITY FEE CALCULATION					FEE
	Number		Number Extra	Rate	Basic Fee $ 690.00
Total Claims		- 20 =		x $18=	$ .00
Independent Claims:		- 3 =		x $78=	$ .00
Multiple Dependent Claims:	N/A	N/A	N/A	N/A	$ N/A
				Total Filing Fee =	$ .00

Enclosed is a check in the amount of $ 1026.00. Please charge any additional fees or credit any overpayment to Deposit Account No. 23-2426 of Winstead Sechrest & Minick, P.C. A **duplicate** copy of this sheet is enclosed.

Please return the original Assignment document to the undersigned attorney for Applicant following recordal of same and address it to: WINSTEAD SECHREST & MINICK P.C., 5400 Renaissance Tower, 1201 Elm Street, Dallas, Texas 75270.

Date _January 25, 2000_
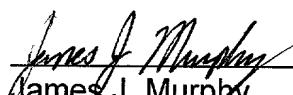
James J. Murphy
Attorney for Applicant
Registration No. 34,503

::ODMA\PCDOCS\DALLAS_1\3256945\1
233:2836-P100US

CERTIFICATE OF MAILING BY "U.S. EXPRESS MAIL"
Under 37 C.F.R. 1.10(c)

"Express Mail" Mailing Label Number ___EM US_570_622_601_US___

Date of Deposit _____ February __/__ , 2000

I hereby certify that this _90_ pages of Application, _5_ pages of Drawings, fully executed Declaration and Power of Attorney, fully executed Assignment, Assignment Cover Sheet and Assignment Recordal fee in the amount of $40.00, Filling Fee Transmittal sheet and check in the amount of $1026.00 application fees are being deposited with sufficient postage with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

S. Taylor Lenz

Signed _____ February __/__ , 2000

CIRCUITS, SYSTEMS AND METHODS FOR INFORMATION PRIVATIZATION
IN PERSONAL ELECTRONIC APPLIANCES

Gregory Allen North
804 Weston Lane, Austin,
Travis County, Texas, 78733

Citizen: United States of America

Matthew Richard Perry
7913 High Hollow Drive
Austin, Travis County,
Texas 78750

Citizen: United States of America

Brian Christopher Kircher
2204 Frontier Trail
Round Rock, Travis County
Texas 78681

Citizen: United States of America

::ODMA\PCDOCS\DALLAS_1\3256948\1
233:2836-P100US

CIRCUITS, SYSTEMS AND METHODS
FOR INFORMATION PRIVATIZATION IN
PERSONAL ELECTRONIC APPLIANCES

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

The present invention relates in general to electronic appliances and in particular to circuits, systems and methods for information privatization in personal electronic appliances.

DESCRIPTION OF THE RELATED ART

Handheld personal electronic appliances have become increasingly popular as new technologies have allowed for the production of affordable devices with a high degree of functionality. One such device is the portable digital audio player, which downloads digital audio data, stores those data in a read-writeable memory, and converts those data into audio on user demand. The digital data is downloaded from a network or retrieved from a fixed medium, such as a compact disk, in one of several forms, including the MPEG Layer 3, ACC, and MS Audio protocols. An audio decoder, supported by appropriate firmware, retrieves the encoded data from memory, applies the corresponding decoding algorithm and coverts the decoded data into analog form for driving a headset or other portable speaker system.

In order to prevent the unauthorized download of copyrighted material, such as music segments, some means of

controlling the operation of a the personal appliance is
desirable. This could be implemented for example through
the issuance of passwords or software kernels authorizing
download the relevant information. The passwords or
5   software must be secured to prevent end user copying,
distribution and tampering. Moreover, the audio decoder may
be operating from proprietary firmware which must also be
protected against copying or tampering.

In sum, what are needed are methods, circuits and
10   systems for securing information in personal digital
appliances. To this end, the ability to secure this
information should not depend on where in the appliance it
is stored, whether it be in memory internal or external to
the primary processing chip. Moreover, implementation of
15   security should not waste resources, such as the available
memory space, which could be used more directly for
processing operations. Additionally, the security methods
and hardware should be applicable to a wide range of
different system configurations.

20   **SUMMARY OF THE INVENTION**

According to the principles of the present invention,
a system is disclosed which includes a central processing
unit operating in response to a set of instructions for
processing information. An interface is included which
25   provides access to selected circuitry forming a part of the
system by an external device. A set of non-volatile
programmable security elements selectively enable and
disable the operation of the interface to provide a private
environment for processing the information. The principles
30   of the present invention provide, among other things the

ability to privatize information in personal digital appliances. These principles can be implemented in a manner which does not waste processing resources, such as available memory space, which could be more directly used for

5   processing operations. Moreover, these principles can be applied to a wide range of different system configurations that do not depend on where in the appliance the private information is to be stored, whether it be in memory internal or external to the primary processing chip.

10   **BRIEF DESCRIPTION OF THE DRAWINGS**

For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, in which:

15   FIGURE 1A is a high level functional block diagram of an integrated circuit embodying the principles of the present invention;

FIGURE 1B is a high level diagram of a second system embodying the inventive concepts;

20   FIGURE 1C is a third exemplary system to which the present inventive principles can be advantageously applied; and 1D are two additional;

FIGURE 2 depicts integrated circuit 100 in a maximum utilization configuration,

25   FIGURE 3 is a high level functional block diagram of the processor depicted in FIGURE 1B;

FIGURES 4A depicts the external clock driving a pin EXPCLK when the clock enable signal on pin CLKEN is asserted with the system entering the Standby State;

5

FIGURE 17 illustrates a preferred procedure for performing an emulated table walk.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The principles of the present invention and their advantages are best understood by referring to the illustrated embodiment depicted in FIGURES 1-17

5   of the drawings, in which like numbers designate like parts.

FIGURE 1A is a high level functional block diagram of an integrated circuit 100 embodying the principles of the present invention. Integrated circuit 100 could be, for example a Cirrus Logic EPxx integrated circuit. Integrated

10   circuit 100 can advantageously be utilized in a number of consumer and industrial handheld information appliances, including personal digital assistants, electronics organizers, and two-way pagers, among other things. In particular, integrated circuit 100 can be configured to

15   perform audio processing in battery powered internet audio decoders.

Two additional exemplary systems to which the present inventive principles can be advantageously applied are shown in FIGURES 1B and 1C and will be discussed further below.

20   FIGURE 2 depicts integrated circuit in a system configuration, and will be referenced during the discussion of the input/output signals (ports) of the various functional blocks of integrated circuit 100.

Integrated circuit 100 is built around an ARM720T

25   processor 101 as described in the ARM720T data sheet available from ARM, Ltd., Cambridge, United Kingdom. Generally, processor 101 includes a central processing unit (CPU) core 102, 8-kilobyte cache 103, memory management unit (MMU) 104 and write buffer 105, each of which will be

30   described in further detail below. It should be noted that in alternate embodiments, an ARM920 processor may also be used.

CPU 102 is a 32-bit microprocessor based on a reduced instruction set computer (RISC) architecture. The associate 8-kilobyte cache 103 is a mixed instruction and data cache (IDC) and is organized as a four way set-associative cache of 512 lines of 16 bytes (4 words).

MMU 104 includes a translation look aside buffer (TLB), access control logic and translation-table-walking logic. The primary functions of MMU 104 are the translation of virtual addresses into physical addresses and the control of accesses to memory. It also supports a conventional two-level page-table structure. Generally, the TLB encaches 64 translated entries and provides the translation to the associated access control logic. If a virtual address causes a hit to a translated entry in the TLB, the access control logic determines whether the access is permitted. In the case of a permitted access, MMU 104 outputs the corresponding physical address from the TLB cache. Otherwise, if the access is not permitted, MMU 104 signals CPU 102 to execute an abort. If a virtual address causes a miss to the TLB cache, the translation-table-walking circuitry retrieves the necessary translation information from a translation table in physical memory. This translation information is written into the TLB cache at a replacement point or entry. The access control logic can then determine whether or not the access is allowed.

Write buffer 105 is used to buffer up to eight words of data and four independent addresses. When enabled, CPU 102 writes data or an instruction into write buffer 105 using an external clock and then returns to instruction execution. Write buffer 105 can then, in parallel, write data onto internal data bus 106 and addresses onto internal address bus 107.

predetermined specific time. RTC 112 is also used to drive a programmable LED flasher (not shown).

Additionally, integrated circuit 100 includes a pair of on-chip timer counters 113. Each timer counter is
5   independent and includes a 16-bit readable-writeable data register. The given counter is loaded to a desired value and then decrements in response to a preselected clock. When the timer counter underflows (i.e., reaches zero) the appropriate interrupt is generated. The timer counter
10 registers can be read at any time. The clock frequency of these timers can be selected by writing to corresponding bits in the system control registers SYSCON. For example, when PLL 108 is sourcing the internal clocks, 512 kHz and 2 kHz rates are available to timer counters 113. When a 13
15 MHz clock from an external source is used, 541 kHz and 2.115 kHz clocks are available. Additionally by using a Divide by 26 circuitry, enabled by setting a bit in system control register SYSCON2, a 500 kHz clock can also be generated from the 13 MHz source.

20   Each timer counter 113 can operate in either a free running mode or a prescale mode by setting or clearing bits in system control register SYSCON1. In the free running mode, the given counter wraps around to 0xFFFF when it underflows (i.e. reaches zero) and continues to count down.
25 In the prescale mode, the value written into the given timer counter is automatically reloaded when the counter underflows. The prescale mode can be used to produce a programmable frequency, drive a buzzer, or generate a periodic interrupt.

30   State control circuitry 114 allows integrated circuit 100 to be set to either an Operating, Idle, or Standby state. A state diagram illustrating the operation of state control circuitry 114 is shown in FIGURE 5. The Operating

state is the normal program execution state and all clocks
and peripheral logic are enabled. The Idle state is similar
to the Operating state with the exception that the CPU clock
is halted pending an interrupt or wake-up to return it back

5   to the Operating state. In the Standby state, PLL 108 is
shut down, although Crystal 111 and oscillator 110 and RTC
circuitry 112 remains active. The external address and data
buses are also forced low in the Standby State to prevent
any powered down peripherals from draining current. It

10   should be noted that integrated circuit 100 when first
powered, or during a cold reset, is forced into the Standby
state, which can only be left by external wake-up prompt.

        In addition to state control circuitry 114, power
management is also effectuated through power management

15   control block 115. The state of various functional blocks
of integrated circuit 100 in each state are tabulated in
TABLE 12.   Power management circuitry 115 forces
integrated circuit 100 into the Standby mode when an active
low power failure signal PWRFL is received from an external

20   power supply unit 201. If integrated circuit 100 is being
driven by an external DC power source 202, the external
power sense input signal EXTPWR is driven active low. If a
battery 203 is being used, an active high on the BATOK pin
indicates that the main battery is OK. The falling edge of

25   this signal generates an FIQ (fast interrupt request) while
a low level signal on this pin in the Standby state inhibits
system start up. The new battery sense signal BATCHG
indicates that a new battery is required; an active low at
this input occurs if the battery voltage falls below a "no

30   battery" threshold. The battery powering integrated circuit
100 could be, for example, one or more standard AA batteries
widely available to retail consumers.

When unexpected events arise during execution of a
program (i.e., an interrupt or memory faults) an exception
is usually generated. When multiple exceptions occur,
interrupt controller 116, operating on a fixed priority
5   system, determines the order in which the exceptions are
serviced.

Integrated circuit 100 operates on two interrupt types,
namely the interrupt request (IRQ) and the fast interrupt
request (FIQ). FIQs have a higher priority than IRQs. When
10   two or more interrupts of the same type occur
simultaneously, the contention is resolved in software.
TABLES 2A-2C set out preferred interrupt allocation, wherein
INTMR1 and INTSR1 are respectively the First Interrupt Mask
Register and First Interrupt Status Register, INTMR2 and
15   INTSR2 are respectively the Second Interrupt Mask Register
and Second Interrupt Status Register, and INTMR3 and INTSR3
are the Third Interrupt Mask Register and Third Interrupt
Status Register. It should be noted that if two interrupts
are received from within the same group (IRQ or FIQ), the
20   order in which they are serviced is preferably resolved in
software.

Generally, interrupt controller 116 operates as
follows. An external or internal interrupting device
asserts the appropriate interrupt. If the appropriate bit
25   is set in the corresponding Interrupt Mask Register, then
either an FIQ or IRQ is asserted by Interrupt Controller
116. If the interrupts are enabled, processor 101 jumps to
the appropriate address. Interrupt dispatch software then
reads the corresponding Interrupt status register to
30   establish the source of the Interrupt and calls the
appropriate Interrupt service routine software which then
clears the Interrupt source through some action specific to
the interrupting device. The Interrupt service routine may

then re-enable interrupts, and any other pending interrupts are similarly serviced. All other external Interrupt sources are held active until the corresponding service routine starts executing.

5   TABLE 3 sets out the external interrupt latencies. In the Operating state, processor 101 checks for a low level on its FIQ and IRQ inputs after each instruction is executed. Hence, there is an Interrupt latency directly related to the amount of time it takes to complete the current instruction

10 after an Interrupt condition is first detected. In the Standby state, the latency will depend on whether the system clock is shut down and if a control bit FASTWAKE in the system control registers is set. As indicated above, PLL 108 is always shut down in the Standby state. If the

15 FASTWAKE bit is cleared, the latency is between 0.125 seconds to 0.25 seconds. If this bit is set however, then the latency will be between 250 microseconds and 500 microseconds. If an external clock is used and disabled during Standby, the latency may also be between 0.125

20 seconds and 0.25 seconds to allow for oscillator stabilization. If the external clock is not disabled, the latency can be reduced to a few microseconds. An Interrupt can also cause integrated circuit 100 to leave the Idle state. In this case the CPU clock must be restarted and

25 additionally, interrupt servicing may be delayed for instruction execution as described above.

    In the illustrated embodiment, an on-chip boot ROM 117 is provided which maintains a set of instructions for initializing integrated circuit 100. On-chip boot ROM also

30 configures UART1, discussed further below, to received 2048 bytes of serial data which are downloaded into on-chip SRAM 118. Once the data has been downloaded into SRAM 118, processor can continue executing instructions by jumping to

the start of the SRAM. Advantageously, this configuration
allows code to be downloaded to program a system flash
memory during the manufacture of a device employing
integrated circuit 100. It should be noted that the user

5   may select between booting from on-chip ROM 117 or from an
external memory connected to port CS[0]. Specifically, if
the signal at pin MEDCHG is low, boot is from on-chip ROM
117 while a high signal applied to this pin requires that
boot be performed from the external memory. It should also

10   be noted that the effect of booting from the on-chip boot
ROM is a reversal of the decoding of all chip select signals
internally. This feature is illustrated in TABLE 5A with
the normal, unreversed chip select decoding illustrated in
TABLE 5B. Additionally, boot can be accomplished from

15   external memory, with the width of the boot device having a
selectable width in accordance with TABLE 4.

The ARM720T processor has a 4 Gbyte address space. In
the illustrated embodiment, integrated circuit 100 uses the
lower 2 Gbytes of the address space for ROM/RAM/Flash and

20   expansion space. Another 0.5 Gbyte is used for DRAM and the
remaining 1.5 Gbytes, less 8K for internal registers, is
unused.

A memory and I/O expansion interface supports six
separate linear memory or expansion segments to external

25   expansion memory 204. Two additional segments are dedicated
to the on-chip SRAM and ROM. Each segment is 256 megabytes
in size. Any of the six segments can be used to support a
conventional SRAM interface. Moreover, each segment can be
individually programed to be 8-, 16- or 32-bit wide, to

30   support page mode accesses, and to execute from one to eight
wait states for nonsequential addresses, and zero to three
for burst mode accesses. The zero wait state sequential
feature allows integrated circuit 100 to interface with

burst mode ROMs. It should be noted that the on-chip ROM space is fully decoded while the complete SRAM address space is fully decoded only up to the maximum size of the video frame buffer used to drive an external LCD (up to 128

5   kBytes).

Two of the expansion segments can be reserved for establishing an interface with two PC Card cards 205 using the chip select signals NCS4 and NCS5. Interface with the external PC cards is preferably made through Cirrus Logic

10   CL-PS6700 PC card slot drivers 206. The memory is segmented to allow different types accesses to take place (i.e., attribute, I/O, and common memory space).

The EXPCLK port to expansion control block 119 outputs an expansion clock which is equal to the CPU clock in the 13

15   MHz and 18 MHz modes, and has a rate of 36.864 MHz when integrated circuit 100 is operating in the 36, 49, or 70 MHz modes.   (EXPCLK port is used as the clock input in the 13 MHz mode discussed above). The EXPRDY pin (Expansion Port Ready) is driven low by the external expansion devices to

20   extend the bus cycle and insert wait states. The chip select signals CS[0:3] are used for SRAM expansions while chip select signal CS[4:5] can be used for either memory expansion or PC card selection. The write strobe WRITE is low during reads from and high during writes to, the

25   expansion devices. The word/halfword bits (2) indicate to the external devices during writes from integrated circuit 100 whether the access size is in words, halfwords or bytes.

DRAM controller 120 provides a programmable 16-bit or 32-bit wide interface to up to two banks 207 of DRAM, with

30   each bank having a storage capacity of up 256 Mbytes. The DRAM banks can be any of a number of types of DRAMs available in the marketplace, including conventional DRAM, synchronous DRAM (SDRAM), extended data out DRAM (EDODRAM),

fast page mode DRAM, and double data rate DRAM (DDRDRAM).
Moreover, these DRAMs can be of the self-refresh type which
are placed in a low power state when integrated circuit 100
enters the Standby state discussed above. To support two
banks, two row address strobes RAS[0:1] can be generated
along with four column address strobes CAS[0:3]. The output
enable signal MOE is used for either the DRAM,
ROM/SRAM/Flash or expansion output enable. The write enable
signal NWE is used for the same set of external devices.
Additionally, DRAM controller includes a programmable
refresh counter, with the refresh period by controlled using
the refresh period register (DRFPR).

The preferred physical to DRAM addressing is shown in
TABLE 6. TABLES 7 and 8 illustrate DRAM address mappings
for 32- and 16-bit DRAM memory systems. The 32-bit is
assumed to be based on two x16 devices connected to each RAS
line with 32-bit DRAM operations selected. The mapping is
repeated for every 256 Mbytes in each bank. The placeholder
"n" is these tables is equal to 0xC + bank number. The
16/32-bit DRAM selection is programmed by setting a bit in
system control register SYSCON2.

Flash interface 121 allows integrated circuit 100 to
interface with flash memory, using the chip select signal
CS[0:1] described above.

LCD controller 122 provides all the necessary control
signals to allow integrated circuit 100 to interface
directly to a single panel multiplexed LCD module 209. The
total frame buffer size is programmable up to 128 KBytes,
using both on and off chip memory. A system can be built
using no external DRAM, with on-chip SRAM 118 used as the
LCD video frame buffer, as described above. The screen is
preferably mapped to the video frame buffer.

LCD direct memory access (DMA) engine 123 is provided for fetching display data for LCD controller 122 from frame buffer memory. The pixel bit rate, hence the LCD refresh rate, can be programmed from 18.432 MHz to 576 kHz when operating in the 18.432-73.728 MHz modes, or 13 MHz to 203 kHz when operating from a 13 MHz clock.

Integrated circuit 100 includes a pair of universal asynchronous receive-transmit (UART) interfaces 124 and 125. These asynchronous ports can be used, for example, to communicate with a pair of RS-232 transceivers 210. Each UART 124/125 can support data rates of up to 115.2 Kbits per second, when integrated circuit 100 is operating from clocks generated by PLL 108. When integrated circuit 100 is being driven from a 13 MHz external clock source, the UART bit rates that can be generated include 9.6 Kbps, 19.2 Kbps, 38 Kbps, 58 Kbps and 115.2 Kbps.

Both UARTs 124/125 include a 16-byte transmit FIFO driving a corresponding transmit (TX) pin and a 16-byte receive FIFO for receiving data from a dedicated receive (RX) pin. An RX interrupt is asserted when a given RX FIFO becomes one-half full or if that FIFO is non-empty for longer than three character length times with no more characters being received. A TX interrupt is asserted whenever the given TX FIFO buffer reaches one-half empty.

In addition to RX and TX ports, UART 124 (UART1) can also receive the three modem control signals CTS, DSR, and DCD. An additional modem control RI input and output modem control signals RTS and DTR can be implemented using the GPIO ports 129 discussed further below. A Modem Status Interrupt for UART1 is generated if any of these modem control bits change.

UART operation and line speeds are programmable through the UART bit rate and line control registers (UBLC1 and

UBLC2). Additionally, four of the FIFOs can also be programmed to have a 1-byte depth. Framming and parity error bits, which are detected at each byte is received, are also readable from 11-bit wide registers.

Integrated circuit 100 also includes an IrDA (infrared data association) SIR protocol post processing stage 126 at the output of UART1 124. It includes a pin for driving an infrared light emitting diode (LED) and an input for connection to a photodiode (shown together as block 211 in FIGURE 2). SRI encoder 126 is switched into the TX and RX ports of UART1, such that these signals can drive the infrared interface directly.

Integrated circuit 100 additionally includes an SPI/ Microwire master mode 128 Kbps ADC interface 127 and serial interface 128, which is shown in further detail in FIGURE 6. A preferred serial pin assignment for the Digital Audio Port is found in TABLE 10. SPI interface 1 (ADC interface) can be used to communicate with an external analog to digital converter 212 and digitizer 213. Serial interface block 128 includes a master slave mode SBI/Microwire (SSI2) interface 603, digital audio interface (DAI) 601, and codec interface 604, all of which are multiplexed through multiplexer 602 onto a single set of external interface pins. The selected interface drives the corresponding circuitry in block 214 of FIGURE 2. Multiplexing is controlled by programming corresponding fields in the system control registers. A summary of the available serial interface options is provided in Table 11.

ADC interface 127 is compatible in a default mode with SSI or Microwire compatible devices such as the MAXIM, MAX148/9 peripherals. ADC interface 127 also can interface with devices such as Analog Devices AD7188/12 chips using NADCCS as a common RFS/TFS line. Exemplary timing diagram

when integrated circuit 100 is driving a MAX148/9 and a AD7811/2 are provided as FIGURES 7A and 7B respectively. An exemplary I^{2}S interface is shown in FIGURE 8.

5   The clock output frequencies for ADC interface 127 can also be set using the system control registers. In the 18.432-73.728 MHz operation mode, the ADC clock (ADCCLK) can be set to either 4, 16, 64, or 128 KHz. When integrated circuit 100 is operating in response to a 13.0 MHz externally generated clock, the ADC clock can be set to 4.2,

10   16.9, 67.7, or 135.4 kHz. The sample clock SMPCLK always runs at twice the frequency of the shift clock (ADCCLK). The available ADC frequency options are set forth in TABLE 12.

           The ADC serial output ADCOUT is fed by either an 8-bit

15   or 16-bit shift register in response to a bit set in the SYNCIO register. The ADC serial input channel ADCIN is captured by a 16-bit shift register. The ACD clock synchronization pulses are activated by a write to the output shift register. During transfers an SSIBUSY

20   (synchronous serial interface) busy bit in the system status flag registers the set. When the transfer is complete and valid data is in the 16-bit read shift register, an SSEOTI interrupt is asserted and the SSIBUSY bit is cleared. The sample clock SMPCLK is independently enabled.

25   Digital Audio Interface 501 provides an interface to CD quality A/D and D/A converters, such as that shown in FIGURE 9. (DAIs are a subset of I^{2}S). 128-bit frames of 16-bit stereo digital audio, at the audio sampling frequency, with separate transmit and receive lines. It should be noted

30   that each frame only contains 16 bits of right channel and 16 bits of left channel audio data. The remaining bits are set to zero.

FIGURE 10 is an exemplary timing diagram illustrating
the operation of DAI 601. The left-right clock (LRCK)
provides the frame synchronization signal. The serial clock
(SCLK) is the bit transfer clock and preferably has a rate

5   fixed at 128 times the audio sample frequency. The SDOUT
(SDATAO) and SDIN (SDATAI) are respectively used for sending
playback data an external D/A converter and for receiving
record data from an external A/D converter. Timing between
integrated circuit 100, an external D/A converter and/or an

10   external A/D converter is based of the oversampled clock
MCLK. Preferably, the MCLK has a rate fixed at 256 times
the sampling frequency.

Asynchronous serial interface 2 (SSI2) 503 is an
SPI/microwire interface that can operate in a full

15   master-slave mode. FIGURE 10 illustrates a pair of
integrated circuit 100 devices configured to operate in a
Master-Slave fashion. The preferred sustained data rate is
85.3 Kbps, which ensures a sufficiently long period between
interrupts. An interrupt is generated when the receive FIFO

20   is half-full and the transmit FIFO is half-empty. In the
slave mode, the serial clock (SSICLK) and the serial receive
port (SSIRXDA), the received synch control pin (SSIRXFR) and
the transmit synchronization pin (SSITXFR) are inputs and
the transmit pin SSITXDA is an output. In the master mode,

25   pins SSICLK, SSITXDA, SSITXFR and SSIRXFR are outputs and
pin SSIRXDA is an input. Mode selection is through the
programming of bits in the system control registers.

Asymmetric (unbalanced) and continuous traffic are both
supported through the use of the separate transmit and frame

30   synch control lines SSITXFR and SSIRXFR. In this
configuration, the receiving node receives a byte of data on
the eight clocks following the assertion of the received
frame synch control signal and the sending node transmits a

byte on the eight clocks following the assertion of the
independent transmit frame synch control pulse. Exemplary
timing diagrams illustrating the operation of these two
interfaces are provided in FIGURES 7A and 7B for reference.

5   Codec Interface 604 supports a direct connection to a
telephony codec. Along with clock and control signal
generation, codec interface 604 also performs parallel to
serial and serial to parallel conversions. The interface is
full duplex and employs corresponding transmit and receive

10   FIFO operating at 64 Kbs. When enabled, the codec interrupt
CSINT is generated every 8 bytes transferred (i.e., FIFO
half full/empty) or, in other words, every 1 msec with a
latency of 1 msec. This timing is illustrated in FIGURE 8,
where CDENRX and CDENTX are respectively the receive and

15   transmit control bits in system control register SYSCON1.

        DAI 601 supports an I^2S interface, such as interface
900 shown in FIGURE 9. In this case, both an external ADC
901 and an external DAC 902. A clock source 903 provides
the time base. An exemplary timing diagram is provided in

20   FIGURE 10. In FIGURES 9 and 10, the MCLK is the oversampled
clock which is typically fixed at 256 times the audio
sampling frequency. The SCLK is the bit clock which is
typically fixed at 128 times the audio sampling frequency.
The LCLK is the frame sync signal and is typically fixed at

25   the audio sampling frequency. SDOUT is the audio data
output sending playback digital audio to DAC 902. SDIN
receives record data from ADC 901.

        SSI1 interface 603 supports master-slave operation as
shown in FIGURE 11. This interface provides a means for

30   effectuating full duplex serial transfers between two nodes.
Data are transferred in bytes in response to a clock and a
frame synchronization signal.

Integrated circuit 100 is also provided with a set of general purpose input output (GPIO) ports 129. In the illustrated embodiment, there are three 8-bit ports and one 3-bit port. The GPIO ports can be used for such purposes as

5   establishing an interface with a keyboard driver 215.

Pulsed with modulator (PWM) circuitry 130 includes two outputs for driving DC to DC 216 converters operating in conjunction with external power supply unit (PSU) subsystem 201. The external input pins normally connected to the

10   output from comparators monitoring the external DC to DC converter output are used to enable these clocks. When integrated circuit 100 is operating from internal PLL 108, the PWM clocks have a frequency of 96 kHz. The duty cycle ratio for these signals can be programmed from 1 and 16 to

15   15 and 16.   The sense of the PWM drive signal active cycle can be set high or low by latching the state of the drive signal during power on reset (i.e., a pull up on the drive signal will results in an active low drive output, and vice versa). As a result, either positive or negative voltages

20   can be generated by the external DC to DC converters. These outputs can similarly be disabled by clearing bits in a control register.

Communication between the blocks of integrated circuit 100 is established through an Advanced Peripheral Bus 132

25   and an Advanced Peripheral Bus Bridge 131. Internal data bus 106 is 32-bits wide and can be connected to the external devices through multiplexing circuitry 132. Internal address bus 107 is 28-bits wide and can communicate with external devices through multiplexing circuitry 133.

30   ICE-JTAG circuitry 134, which is IEEE 1149.1 compliant, is included for boundary scanning during test and development. Additionally, the Embedded ICE supports the debugging of the ARM processor core.

In the preferred embodiment, the internal registers of integrated circuit 100 are in the little endian configuration. However, integrated circuit 100 can advantageously interface with a big endian external memory system. Specifically, the big end bit and the CPU 101 register sets determines whether words in the external memory are being stored in a big endian or little endian format. Specifically, memory is viewed as a linear collection of bytes numbered upwards from zero. Bytes 0-3 hold the first stored word, bytes 4-7 the second stored word, and so on. In the little endian scheme, the lowest number byte in a word is considered to be the least significant byte of the word and the highest number byte is the most significant word. Thus, byte zero in a little endian system are connected to data lines 7-0. In the big endian scheme, the most significant byte of a word is stored at the lowest numbered byte, and the least significant byte is stored at the highest number byte. Therefore, byte zero in a big endian system is connected to data lines 31-24. In the illustrated embodiment, only the load and store instructions are effected by endian-ness. TABLES 13 and 14 illustrate the operation of integrated circuit 100 for both reads (TABLE 13) and writes (TABLE 14). It should be noted that the column address strobe lines NCAS[3:0] to the DRAM banks are always connected to the same byte lane irrespective of the endianness. For example, column address strobe line NCAS[0] will be associated with data line D[7:0] and NCAS[3] associated with data lines D[31:24]. As a result, in a little endian system, line NCAS[0] will be asserted for a read/write to DRAM byte 0 and in a big endian system, line NCAS[3] will be asserted to access DRAM byte 0.

Integrated circuit 100 includes a set of programmable fuses which allow each chip to be assigned one or more

unique ID numbers and passwords. The programmable fuses and related registers are disposed within Security Registers and Hardware block 133 operating off APB 132 (FIGURE 1). With specific regards to the embodiment of FIGURE 1D, the boot

5 ROM itself will reside on the ARM local bus 107 and the access checking will be split and have logic on both the ARM local bus and within the ARM local-global AHB wrapper.

In the preferred embodiment, the are 256 programmable fuses, including sets of public and private fuses. The

10 addresses and values of the private fuses are hidden such that only private firmware corresponding to those fuses is allowed accesses. In a non-private environment, these addresses and values return all zeros. The public fuses are described in TABLE 15 and the private fuses in TABLE 16 .

15 Integrated circuit 100 also includes embedded hardware within block 133 to check the fused hamming code with the hamming code that matches the selected ID. When the validation address is read, the ID value is matched with it hamming value and verified. The resulting 5 bit code

20 provides debug information (all fuses blown or all fuses not blown) if the hamming codes do not match. Table 17 gives the decoding for the validation read bits. This advantageously allows detects errors which have occurred when the fuses were blown, while preserving the

25 inaccessibility of the fuse values and addresses.

Table 18 provides the addresses that return the validation codes for the public ID-CHK pairs.

Table 19 provides the addresses that return the validation codes for the private ID-CHK pairs. These

30 addresses are only accessible by the firmware when integrated circuit 100 is operating in a private mode and will read 0's otherwise.

In order to allow the hamming code generator to be adequately tested there are two test registers that can be selected as the ID-CHK pair and validated. Their definitions and locations are provided in Table 20.

Figure 1B is a high level functional block diagram of a second system on a chip 140 suitable for practicing the principles of the present invention. This embodiment employs an ARM920T processor 141 having both instruction and data caches, as well as an MMU. System 141 does not include general purpose SRAM in contrast to integrated circuit 100.

FIGURE 13 is a more detailed functional block diagram of processor 141, in particular for those embodiments based on an ARM 920T core. In this embodiment, the available cache comprises both an instruction cache 1301 and a data cache 1302. Similarly, separate instruction and data MMUs 1303 and 1304 are used. The instruction modified virtual address (IMVA), instruction physical address (IPA) and instruction data (ID) buses are each 32 bits wide. Similarly, the data modified virtual address (DVMA), data physical address (DPA) and data data (DD) buses are 32 bits wide. Physical addresses and data are exchanged to AHB bus 142 through AMBA bus interface 1305. A write buffer 1306 allows for the parallel exchange of data through interface 1305 during processor core operations. Data from data cache 1302 can be output through write-back physical address (PTAG) RAM 1307.

Integral to the processor core is a coprocessor which includes a register for translating virtual addresses issued by the CPU into the modified instruction and data virtual addresses (MVA) transmitted on IMVA and DMVA shown in FIGURE 1B. Specifically, for addresses to the memory region from 0 to 32 MBytes, the virtual address VA is modified by a 7-bit process identifier as VMA = VA + (ProcID x 32 MBytes), where

the process identifier ProcID is either a read or write process identifier.

System 141 is based on an internal AHB (Advanced Microcontroller Bus Architecture High-Speed Bus) 142, as well as an internal an APB (Advanced Peripheral Bus) 143. An AHB/APB bridge 144 interfaces AHB 142 and APB 143. A second bridge 145 interfaces processor 141 with AHB 142.

Among the devices operating off AHB 142 are Graphics Engine 146 and Raster Engine 147. Generally, Graphics Engine off-loads such functions as block transfers and line draws from processor 141 to improve system graphics performance. Preferably, Graphics Engine 146 uses a standard Device Independent Bitmap (DIP) format for supporting Windows CE. Raster Engine 147 is provided to raster data from an external display buffer, through synchronous DRAM interface 148, to drive an external LCD, CRT or TV display unit.

Additional on-chip interfaces to internal AHB include an interface 149 for coupling system 141 to an external system bus, a PCMIA for interfacing with an external PC card, and Test Interface Controller (TIC) interface 151 for testing such on-chip circuit blocks as the DMA controller and the raster system. Memory interface 152 provides for the exchange of control signals and data with external SRAM, Flash or ROM in a manner similar to that discussed above. Finally, boot of the system, which will be discussed further below, is effectuated, at least in part, using the Boot ROM 153. In this example, boot ROM 153 is operating off AHB 142, although it could run off any one of a number of global and local buses in alternate embodiments.

System 140 includes an 8-channel DMA engine 154, which prioritizes and services request by on-chip resources, such as the UARTs, for accesses to external memory. The Joint

Test Action Group (JTAG) port 155 supports debugging of the on-chip processor and related circuitry Additionally, a Universal Serial Bus (USB) controller 156 and Ethernet port 157 operate directly from the AHB.

5   A number of peripheral devices are provided on-chip and operate off of APB 143. Among other things, system 140 includes three UARTs 158, 159 and 160. Additionally, a pair of SP1 interfaces 161 and 162 and an AC97 interface 163 are included in the illustrated embodiment. A real time clock

10   (RTC) 165, general timer set 166 and watchdog timer 167 are also provided in this embodiment. An additional memory interface, EEPROM interface 168, also couples to the APB.

Manual input of data can be made through an external key matrix coupled to Key Matrix Interface 169, or a

15   Touchscreen interfacing with Touchscreen ADC 171 and Touchscreen Interface 170. LED outputs 172 are also included in the system 140 user interface.

Similar to integrated circuit 100, system 140 includes a set of general purpose input/output (GPIO) ports 173, an

20   interrupt controller 174, and on-chip PLLs 175 driving system control circuitry 176. Control circuitry includes memory remap and system pause control circuitry 177. Flash VPP Control block 178 generates the voltages required for writing and erasing external flash.

25   FIGURE 1C is a high level functional block diagram of another exemplary system-on-a-chip 180 to which the principles of the present invention can be suitably applied. In this instance, the CPU core 181, which could be for example a ARM7TDMI controller, does not utilize an MMU or

30   on-chip cache. CPU 181 operates in conjunction with the AHB bus 142 via a local AHB bus and Local/Main AHB interface 182. CPU 181 is supported by memory 183, security gates 184

and security/reset circuitry 185. Security will be discussed in further detail below.

In this embodiment, system 180 additionally includes a digital signal processor (DSP) 186 supported by global memory 189, data memory 190 and program memory 191. Interprocessor communications registers 192, I^2S Audio Input/Output Port 193, PWM circuitry 194 that is capable of driving external speakers at CD quality levels without using analog DAC support circuits, and DSP Timer/STC 195 communicate with DSP 186 via DSP Peripheral Bus 196. These devices also operate off the APB. Among the peripheral devices also operating from the APB include a USB Slave Port 197, SPI for Serial Media Input 198 and I^2S Host Port 199.

The Motion Picture Expert Group (MPEG) audio compression standard defines the syntax for a coded stream of digitized audio data, along with a process for decoding that stream. In the audio arena, three layers, Layers I - III respectively, are defined. For purposes of the present discussion, Layer III, which provides the highest quality audio reproduction, will be considered.

The encoding process begins with the sampling of one or more audio channels at a given sampling rate, which may be 32, 44.1 or 48 kHz. The resulting digitized stream is passed through a polyphase filter bank which divides the received time-domain stream into 32 frequency subbands. Typically the filter bank operates of 64 input samples at a time with 50% overlap such that 32 output frequency-domain samples are produced for 32 input time-domain samples.

A psychoacoustic model is used to remove those parts of the audio signal which cannot be heard by the human ear due to auditory masking. Auditory masking is the characteristic of the human auditory system wherein a strong audio signal renders a temporally or spatially close weaker audio signal

imperceptible. Moreover, the ability of the human ear to
distinguish sounds is frequency dependent. Within certain
critical bands, the ear does not precisely delineate between
various in-band audio components. The processing subbands,

5   which approximate these critical hearing bands, are
quantized as a function of the audibility of the
quantization noise within that subband.

       The psychoacoustic model engine, operating in parallel
with the polyphase filter, determines the available noise

10   masking for a given frequency component and a given
loudness. From this information, the data stream output
from the polyphase filter are quantized and coded. In Layer
III, each of the 32 subbands output from the polyphase
filter are passed through a window which parses the stream

15   into long blocks of 18 samples or short blocks of 6 samples,
with 50% overlap such that the window lengths are
respectively 36 and 12 samples wide. Long blocks are used to
achieve better frequency resolution for the relatively
constant components of the audio signals while short blocks

20   are used for improved frequency resolution of transients.
The blocks for each subband are then processed with a
Modified Discrete Cosine Transform (MDCT). The subbands are
further divided in frequency to improve spectral resolution
such that some of the aliasing caused by the polyphase

25   filter can be canceled.

       In MPEGx, Layer III, the quantization is non-uniform to
make the signal to noise ration over the range of
quantization values more consistent. Additionally, Layer
III utilizes scale factor bands of approximating critical

30   band widths and cover several MDCT coefficients. The scale
factors are used during noise allocation to vary the
frequency-dependent masking threshold, and essentially set
the gain for each subband. Moreover, Huffman encoding is

performed on the quantized MDCT coefficients for improving
data compression. Finally, a "bit reservoir" is employed,
to which bits can be donated when less than the average
number of bits are required to code a frame and from which
bits can be borrowed when more than the average number of
bits are required to code a frame.

Frames are formed from a header, a CRC value, side
information and main data, although the relative position of
these components of the frame are not necessarily always in
the same sequence, or even adjacent in the stream. The
header includes a set of frame sync bits, MPEG version and
layer identifiers, a CRC protection bit, a bitrate index
indicating the bitrate at which the frame was created, and a
sampling rate frequency index indicating the frequency at
which audio data was sampled, and along with additional
information about the transported data.

An MPEG-1, Layer III bitstream can then be decoded
generally as follows. Data is input to the decoder in a
predetermined number of frames per second. The frame sync
bits in the header portion of each frame is detected. Next,
the scale factors are extracted and decoded. This is
followed by decoding of the Huffman encoded main data
representing the frequency energies. The scale factors are
applied and the data requantized. At this point, if stereo
data is being processed, the stereo channels are recovered
and aliasing reduction performed. An inverse MDCT operation
is performed to followed by an overlapping inverse Discrete
Cosine Transform (DCT) to return the data to the
time-domain. A low pass filter is applied to recover the
PCM samples, each of which is essentially a weighted average
of the adjacent 512 time-domain samples.

When integrated circuit 100 is configured as an MPEGx,
Layer III decoder, a stereo DAC, such as a Cirrus Logic

CS43Lxx Stereo Audio DAC, is coupled to Digital Audio Port 128 for driving a set of headphones. An analog to digital converter, such as a Cirrus Logic CS53L32 Audio A/D Converter may also be coupled to this port for the input of data from a microphone. This embodiment of FIGURE 1D includes an on-chip PWM circuit that can drive headphones at CD quality levels without an external stereo audio DAC.

It is often necessary to prevent tampering, copying or logic analyzer examination of the software and firmware bundled with an electronic product. As a result, some level of security must be provided, for example through the use of encrypted passwords, which allow the manufacturer authorized end users access to the system memory assets for purposes of downloading, debugging, and upgrading the software or firmware, but denies that same level of access to unauthorized end users. In the context of a digital audio player, this will allow online music distributors the confidence to allow end users who have paid the royalty and received the requisite passwords to download songs, with the knowledge that unauthorized downloads will at least be deterred to some degree.

Generally, there are several criteria a security scheme must meet. First, the system must not allow unauthorized access as a result of power-on reset. Second, secure information, such as encrypted passwords, security code, and the information concerning locations in memory where the secure information resides, must not be readily accessible outside the system. Notwithstanding, this secure information must be checked during production test procedures to guarantee acceptable end user system quality with regards to normal manufacturing defects. Finally, if security measures are not provided or not invoked, normal

operation of the system should proceed in the expected fashion.

Advantageously, the principles of the present invention provide security techniques which allow integrated circuit

5   100 to meet each of these criteria. In accordance with one such technique, the capability of processor 101, in response to either certain default conditions or the dynamic assertion of certain instructions, to reverse the Chip Select signal decoding discussed above is employed. By

10   reversing the chip select decoding on power-on reset, the security code can be run from a normally unaccessible memory space. Moreover, this feature of processor 101 can only be invoked in a specific period of time when processor 101 is not executing instructions, which further complicates any

15   attempted security breach.

FIGURE 12 is a flow chart illustrating a preferred procedure 1200 for booting integrated circuit 100 in accordance with the inventive concepts. It will be assumed that processor 101 is an ARM720T or ARM920T processor, and

20   signal names will be in reference to the signals and/or instructions thereof. The procedure begins with the power on reset of integrated circuit 100 by the assertion of the power-on reset (NPOR) signal, at Step 1201. Circuits within the system immediately disable all hardware and debug

25   features and hides all security elements (e.g. firmware, registers, passwords) from external probing (Step 1202). This step insures that the system is secure, at least up until Step 1203, where a check is made to determine if security firmware routines are in place and enabled. In the

30   preferred embodiment, this is accomplished by reading the programmable fuse registers.

For purposes of discussion, the case where security is not provided or is disabled will be considered first. At

Step 1204, a determination is made as to whether boot is to
continue from an internal ROM or if an external memory will
be used. For the ARM processor embodiments, the NMEDCHG bit
is used to select between internal and external boot memory

5   options. If at Step 1204, the signal at pin NMEDCHG is
clear (i.e. in an active low state), then boot of integrated
circuit 100 will be from internal ROM. In this case, the
address mapping to internal boot ROM is reversed by default
at Step 1205. After reversal of the address mapping,

10   execution is from current boot ROM location 0 (Step 1207).
In this illustrated embodiment, the power-reset signal NPOR
must be asserted to return the address mapping to its normal
state.

       Alternatively, if the NMEDCHG bit is set (i.e. in an

15   active high state), then boot will be from external memory
(ROM/EPROM/Flash). In this case, the chip select mapping is
set as shown in TABLE 5A with the external Chip Select 0
being selected as the boot memory.

       Next, consider the case where a read of the

20   programmable fuse registers indicates that a security
routine is in place and enabled. Boot branches at Step 1208
to the execution of the security procedure.

       Integrated circuit 100 can be configured to respond to
different sets of boot and/or security code. This

25   advantageously allows integrated circuit 100 to operate
using the boot/security firmware from multiple vendors, even
though the secure information of each vendor may only be
accessible by that vendor's own boot/security procedures.
Initially, the boot memory is programmed with multiple boot

30   code sets or options. This can done using the internal boot
ROM or one or more chips of external memory (ROM/RAM/Flash).
With multiple boot options, the end user will be able to

select between security firmware available from different
vendors.

Consequently, at Step 1209 a first one of the booting
options in boot memory is identified and at Step 1210
aliased to the reset vector, typically location 0x00 for the
first option. All necessary security elements (registers,
firmware, I/O devices) required for the given implementation
are enabled by the current boot option while all other
security options (implementations) are kept hidden (Step
1211). The selected boot code is then run by the processor
at Step 1212 to attempt to initialize for the selected
security firmware/software.

If at Step 1213 the proper security firmware/software
is found in memory as called by the boot code, then
integrated circuit 100 completes boot and runs in the
selected secured environment at Step 1214 under supervising
control of the security firmware/software. On the other
hand, if the required security firmware/software is not
found, another boot option must be tried.

If the last security option has not been reached at
Step 1215, then the next security option in boot code is
selected (Step 1216). An instruction is issued which
dynamically forces the processor to the new reset vector.
In this instance, the reset vector jumps to point to the
second security option in boot code. At Step 1218, the
processing returns to Step 1211 and the boot process is
attempted again. It should be noted that in the illustrated
embodiment, the instruction pipeline has three stages.
Consequently, the instruction resetting the program counter
to 0 has already been loaded from internal boot ROM before
execution of the instructions that changes the chip selects.
The MOV pc, #0 instruction causes the processor pipeline to
be flushed thereby allowing several cycles to occur before

the change of chip selection must occur. During this process, no other accesses are allowed to those memory resources whose chip select signals will change during the execution of the remap command.

5   This process repeats itself until either a security option is found which causes integrated circuit 100 to enter secure operation at Step 1214 or the last security option is reached at Step 1215. In the illustrated embodiment, the last or default option returns the boot procedure at Step

10   1219 back to a normal (unsecure) boot. Here, all the debug features are enabled and the security features are hidden at Step 1220. At Step 1221, a default boot ROM is selected and at Step 1222 the processor is dynamically forced to the reset vector. Notwithstanding, in alternate embodiments,

15   default security code may be provided in order that integrated circuit 100 can still run in a secure environment even though all of the primary options are unavailable.

     In embodiments of integrated circuit 100 based upon an ARM 920T, instructions and data can be locked into the

20   corresponding instruction and data caches, such that they are not chosen as victims for replacement by the replacement algorithm on a cache miss. Locked in data/instructions guarantee a cache hit with the corresponding information being fetched directly from cache and the favorable cache

25   access latency. Moreover, the locked encached information is unaccessible outside of integrated circuit 100, except through the JTAG port or other test-debug modes that allow visibility to the cache or TLB memories. The JTAG port, used primarily during product development and testing, can

30   be disabled integrated circuit 100 leaves the manufacturing floor.

     Before locking cache entries, the corresponding descriptors (physical addresses and permissions) must be

locked in the associated Translation Look aside Buffers
(TLBs) to obtain predictable performance results.   In
addition to cache, many devices, such as the ARM 920T used
in the present examples, include both data and instruction

5   translation look-aside buffers (TLBs).   For a given
instruction or field of data, the CPU generates a virtual
address.   A modified virtual address is then presented to
the corresponding TLB and a comparison is performed between
fields of the modified virtual address and the comparison

10   (tag) registers in the TLB.   If a match occurs, and the
access is allowable,   (as determined by the permission
fields in the TLB entry), the physical address bits returned
from the corresponding TLB entry is used, along with index
bits from the modified virtual address, to generate a

15   physical address, accessing cache or external memory, as
required.   If a miss occurs, the process discuss below is
invoked to translate the virtual address into a physical
address in hardware.

       When cache lines are locked, the corresponding entries

20   in the data and instructions TLBs should also be locked and
exempted from replacement during TLB update.   In the case of
the ARM 920T processor, TLB entries are locked by writing
identifiers for the specific entries in the data and
instruction TLBs being locked into TLB Lock Down field of

25   the System Control Processor register C15.

       TLB Lockdown procedure 1300 of FIGURE 13 is one method
of locking entries in either an instruction or data TLB.   At
Step 1301, a page table is setup including physical address
bits and permissions corresponding to the protected data or

30   instructions.   At least some of the entries in the target
TLB are then flushed or cleaned to insure that the code to
be locked is not already in the TLB registers (Step 1302).

In embodiments employing the ARM 920T processor, both the data and instruction TLBs are organized in a single segment of 64 lines. A replacement (victim) counter points to the entry being replaced. Therefore, at Step 1303, the replacement counter is updated to point to the next entry to which locked information is to be written. In the preferred embodiment, the process begins at entry 0.

For the instruction TLB a Prefetch instruction is used to generate a modified virtual address force a TLB miss (Step 1304). In the case of the data TLB, a Load instruction can be used to force the miss. After the miss has occurred, a page table walk must be performed to generate the descriptor (e.g. physical address and permissions) to be loaded into the TLB (Step 1305). At Step 1306, the descriptor generated from the page table walk, using physical address bits from the accessed page table entry and index bits from the modified virtual address, is loaded into the given TLB at the entry pointed-to by the current replacement counter contents.

In the ARM 920T embodiments, the loaded TLB entry is locked at Step 1307 by setting a bit in a corresponding TLB Lockdown register. If the last entry has been reached at Step 1308, the procedure ends, otherwise, at Step 1309, the procedure loops back to Step 1303 and the replacement counter updated in preparation to load the next entry.

Once the TLB entries are locked, the corresponding data or code can be locked in cache. For purposes of discussion, consider the case where instructions are locked into the ARM 920T instruction cache. The case of the data cache is similar. It should also be noted, that the inventive concepts are not limited to systems employing ARM processors, and can be applied to any system or device including Lockable instruction and/or data cache.

FIGURE 14 illustrates a cache lockdown procedure 1400 for locking secure code into cache. As will be discussed further, in order to perform the locking operation, a cache miss must be forced in the illustrated embodiment. A preferred method of forcing a cache miss is discussed later in conjunction with FIGURE 15.

At Step 1401 an actual or emulated page table is set up with the physical addresses to the locations in memory where the data or instructions to be locked in cache reside. An emulated and synthesized page tables embodying the inventive concepts are also discussed further below. This table is used to update the corresponding TLB, preferably using procedure 1300.

At Step 1402, the given cache is flushed or cleaned of at least some cache lines to insure that the code to be locked-in is not already encached. The replacement (victim) counter associated with the cache is forced to point to the first cache line (cache line 0) at Step 1403. In the preferred embodiment, each of the data and instruction caches is partitioned into 8 64-line segments each indexed by index fields in the modified virtual address. In procedure 1400, cache lines will be filled sequentially, with for example, all cache lines 0 of all segments filled in sequence first, followed by the sequential fill of all cache lines 1, and so on.

The data or instructions to be encached are generated, possibly requiring a decryption process (de-encrypted), and stored at corresponding locations in an alternative memory, such as internal SRAM or external SRAM/DRAM/Flash at Step 1404. Then, at Step 1305, a Prefetch Cache Line operation is performed for an instruction encache to invoke a look-up at the pointed-to cache entry. (For the data cache, the LOAD instruction can be used). This causes a cache miss

thereby requiring the processor to access the alternate memory containing the necessary data or instructions. It can do this by referring to the TLB for the necessary bits for the physical address, if the TLB is current and accurate, or by walking through the page tables set up at Step 1401 directly. The physical address itself is generated from base address bits in the entry accessed in the TLB and index bits from the virtual address.

At Step 1405, the generated code or data is placed where the cache miss is to be processed and a line fill is performed at Step 1406 to the cache line at the current replacement pointer entry. Again, the cache segment is indexed by cache segment index bits from the virtual address causing the cache miss.

If, at Step 1407, the last segment in the given cache not been reached and further cache operations are required, the processor increments the cache segment index bits at Step 1408 to force the next cache access to the next cache segment at the current replacement counter value. The procedure returns to Step 1404 and continues from there. However, if the just completed operation was to the last cache segment, and more cache operations are to follow (i.e. the last cache line to be filled has not been reached at Step 1409), then at Step 1410, the procedure jumps back to Step 1403, the replacement counter value is updated, and the procedure continues from that point.

When all the code to be locked-in has been loaded, the Replacement Counter base is set to a value one higher that the base to the locked cache lines (Step 1411). This insures that the private data (now decrypted) will not be overwritten on a cache miss or become accessible by an unauthorized party. The code can then be executed from cache at Step 1412.

One means of creating locked, encached data without using memory locations for the entire region to be locked is to use a cache line's length of registers to emulate the region. In addition, cache miss emulation can also be used

5   to remedy hardware limitations on the cache locking granularity. For example, in the ARM 920T embodiment, cache can be locked in 64 word blocks (256 bytes). Each cache line however is only 8 words (32 bytes) long, and therefore can be mapped a different locations within the 64 word

10   block, depending on the address bits.

According to the inventive concepts, for each lockable location, eight (8) programmable 32-bit emulated cache line (ECLINE) registers are set up in an alternate location in memory as eight contiguous 32-bit locations. Additionally,

15   a comparison (offset) register (ECOFFSET) is setup which is programmed with physical addresses to identify where in the cache memory space the ECLINE register contents will reside after the emulated cache miss. Consequently, the location a single cache line in size can be used to represent an entire

20   64 word Lockable location.

An emulated cache miss procedure 1500 is then set forth in the flow chart of FIGURE 15. At Step 1501, the contents to be encached (in either the instruction or data cache) are written into the ECLINE registers. An offset to the

25   Lockable cache space to which the data are to be written is then programmed into the ECOFFSET comparison register (Step 1502).

At Step 1503, an operation is performed to cause a cache miss For the instruction cache, this can be done

30   through a Prefetch Instruction for the instruction cache, and for the data cache, through a Load. The virtual address generated to this location causes a miss to the given cache, and the corresponding physical address is then generated

using index bits from the virtual address and base bits
retrieved from the appropriate TLB or alternatively, through
a page table walk. At Step 1505, the information in the
corresponding ECLINE registers is retrieved and at Step

5   1506, loaded into the cache at the addressed entry. This
entry is now prepared to be locked using procedure 1400.
Advantageously, the procedure has allowed the locked portion
of cache to be loaded without resort to either internal or
external SRAM.

10   As already indicated, page table walks are required
during cache and TLB locking operations in order to generate
addresses to physical memory from where the data or
instructions are to be retrieved. The present inventive
concepts allow for the creation of streamlined page tables

15   which save on the amount of memory which must be dedicated
to page table support. Moreover, even in view of a TLB
miss, the inventive concepts also protect data and
instruction code against tampering, copying or electronic
analysis through secure operation of MMU 104 during address

20   translation by section/page table walks. Again, an ARM 920T
processor core will be considered for illustrative purposes,
although the inventive principles can be applied to the
memory management schemes of other processors and memory
management units.

25   A conventional page table walk for this embodiment
generally proceeds as follows. During the Level 1 fetch, a
section descriptor (Level 1), course page table or fine page
table base address is retrieved from a 4096-entry
Translation Base Table (TBT). The TBT is accessed using a

30   TBT base address from the Translation Base Register and a
Table Index field from the modified virtual address.

          If the output from the TBT is a section descriptor,
that descriptor includes a Section Base Address and access

permissions. A physical address to a 1MByte section of memory is then generated using the Section Base Address bits from the Level 1 descriptor and Section Index bits from the modified virtual address. (Assuming that the permissions

5 contained in the Level 1 Section Descriptor are favorable).

A Course Page Table Base address from the TBT, along with Level 2 Table Index bits from the modified virtual address, accesses one of 256 entries in the Course Page Table thereby dividing the 1 Mbyte blocks into 4 Kbyte

10 blocks. The Course Page Table returns either a Large or Small Base Address along with access permissions. Depending on the state of the permissions, the Large or Small Page Base Address bits are combined with Page Index bits from the modified virtual address to produce a physical address

15 either a 64 Kbyte large page or 4 Kbyte small page from memory.

A fine page base address from the TBT, along with Level 2 Table Index bits from the modified virtual memory address, points to a 1024 entry Fine Page Table. The output from

20 this table is a Level 2 Descriptor which includes either a large, small or tiny base address along with access permissions. Large pages are 64 kBytes, small pages 4 KBytes and tiny pages 1 Kbyte. Assuming the permissions indicate that the access is allowed, the page base address is

25 concatenated with Page Index bits from the modified virtual address to generate a physical address to either the large or small pages in memory already discussed, or 1kbyte tiny pages in memory.

The memory accessed as a result of the page table walk

30 can be either cache, internal memory or external memory. The physical addresses and permissions are used to update the TLB. Any secure information is then locked, as described above, into the TLB.

The disadvantage of this two-level table walk procedure comes from the fact that the various tables require substantial amounts of on-chip memory. As previously discussed, secure information must reside inside the system in areas of the memory not accessible by an unauthorized user. Therefore, some provision must be made for efficiently storing sensitive information, such as the physical address translation scheme, within the available internal memory.

In the preferred embodiment of integrated circuit 100, the table walking process can be significantly simplified and the amount of memory required for the translation tables greatly reduced. Not only is this important in terms of increased operating efficiency, but it also insures that resort to unsecure external memory is not required.

Here, the memory space is divided up into 256 MByte regions, each of which is associated with a common set of access characteristics.(e.g. access permissions, cacheability, bufferability). Of only one of these regions, only 1 MByte requires a second level page table. Thus, since large regions of memory have common access characteristics, much smaller translation tables can be created within the available SRAM space.

The access permissions indicate whether given information can be accessed from the corresponding memory block. The cacheability and bufferability attribute bits are used to determine if an accessed piece of information can be stored in cache or transferred through the write buffer. For example, the contents of the real hardware registers controlling the UARTs and other peripherals and I/O devices are generally not allowed to be cached or buffered by the CPU subsystem. This would cause incorrect

behavior of these peripherals due to the timing of when the accesses would actually occur.

Moreover, for a secure system the page/section table information must be kept within the confines of the private area such that this information can not pass from memory to the device pins that can be examined by a logic analyzer.

In the illustrated embodiment, where the memory is divided into 16, 256 Mbyte blocks, a 32-bit register is created for storing the Level 1 AP bits, each two bit pairs mapping to a 256 MByte memory region. For example bits [1:0] map to Region 1, bits [3:2] to Region 2, and so on. A 16-bit register is setup for holding a set of bits indicating the cacheability of each region for Level 1. Another 16-bit register is setup for holding either a set of bits indicating the cacheability of each region. These registers are pointed to by the contents of the Translation Base Register in the MMU.

A procedure 1600 for updating these registers, as well as for handling memory regions which have unique characteristics and constants, is shown in FIGURE 16A.

For a given 256 Mbyte region, a determination is made at Step 1601 as to whether it has a common set of access characteristics. If the determination is affirmative, then at Step 1602, the corresponding entry in the global Level 1 AP register is loaded with the appropriate AP bits. The corresponding entries in the global Level 1 bufferability and cacheability registers are similarly updated at Steps 1602 and 1603.

At Step 1605, the procedure returns for the update of the register entries for the next memory region (block) requiring update. Initialization/update of the global access control registers is preferably done in a loop. The values in general do not change but can be updated if

necessary during system processing. The full register
values for entries that are not to be synthesized are
updated as appropriate during system operation. For example
they will need updated when a page of memory is substituted
for another when it is "swapped" out to disk or similar mass
storage devices.

For those memory blocks or registers which have a
unique set of access characteristics at Step 1601, including
access permissions, bufferability and cacheability bits, and
physical address bits, a full 32-bit register is loaded with
a complete Level 1 descriptor at Steps 1606 and 1607. The
procedure again loops back at Step 1608. This descriptor
can include a course or fine page (Level 2) table address.
Otherwise, a constant is pointed to in hardwired gates at
Step 1608. The stored constant can be a fixed value or a
base address to a Level 2 table. If a walk to Level 2 is
not required at Step 1609 then the procedure loops back at
Step 1610. Otherwise, at Step 1611, a corresponding
register in the Level 2 synthesized table is set up at Step
1611.

A similar process is used to synthesize Level 2.
Specifically, for each Level 2 page, a register pointed-to
by the Level 2 base address bits from the Level 1 registers
A global Level 2 AP register, along with Level 2
bufferability and cacheability registers are set up as
before, for pages and sub-blocks having common
characteristics.

For a given page or set of pages, a determination is
made at Step 1612 as to whether it has a common set of
access characteristics. If the determination is
affirmative, then a Step 1613, the corresponding entry in
the global Level 2 AP register is loaded with the
appropriate AP bits. The corresponding entries in the

global Level 2 bufferability and cacheability registers are similarly updated at Steps 1614 and 1615.

At Step 1616, the procedure returns to Step 1601 for the update of the register entries for the next memory region (block) requiring update.

For those Level 2 pages, sets of pages, blocks or registers which have a unique set of Level 2 access characteristics at Step 1612, including access permissions, bufferability and cacheability bits, and physical address bits, a full 32-bit register is loaded with a complete Level 2 descriptor at Step 1618. Otherwise, a constant is pointed to in hardwired gates at Step 1618. The stored constant can be a fixed value, base address, or the like. The procedure again loops back at Step 1619 to Step 1601.

An exemplary synthesized page table walk is illustrated in FIGURE 12B. At Step 1620, the table walk is requested. This request could be in response to a TLB and/or cache miss. In this example, consider first the case where a second level of Table walk is not required at Step 1621. The Level 1 registers discussed above are then pointed-to by the translation base register in the MMU at Step 1622. The Level 1 register entries are indexed using the table index bits from the virtual address (Step 1623).

At Steps 1624 and 1625, a determination is made as to whether the return from the indexed entry in the Level 1 registers is either a full descriptor or a constant. The case in which the return is neither a constant nor a full descriptor will be considered first.

At Step 1626 the access control bits in the first level global access registers (i.e. the AP, cacheability and bufferability bits) are retrieved. The table index from the virtual address are then transformed into physical address

bits at Step 1627 by moving bit positions relative to the virtual address.

In the preferred embodiment, the transformed virtual address bits for section entries will be the table index bits (bits 13:2 of the lookup word index into the 4096 entry level 1 page table) become bits (31:20) of the result for the entry (1 Mbyte memory region). The domain of the section will be defined by bits (13:10) of the memory location. For embodiments using the ARM920 or ARM720 MMU), several bits in the page table entries are always a constant 0 or 1.

The Level 1 descriptor is formed at Step 1628 by merging the transformed address bits and the retrieved access control bits. The synthesized descriptor is returned at Step 1629 for updating the TLB and or cache.

Returning to Steps 1624 and 1625, the Level 1 entry can also be a full descriptor (Step 1630), or a constant (Step 1631). The descriptor or constant can be used immediately at Step 1632.

Assume next that a Level 2 table reference is required at Step 1621.

The Level 2 translation is similar to that performed when only a Level 1 reference is required. At Step 1633, the Level 2 registers, setup as described above, are pointed to by a base address in the MMU. The specific register or entry is indexed using the table index bits from the virtual address at Step 1634. A determination is made at Steps 1635 and 1636 as to whether the indexed register (entry) contains a full descriptor or a constant. If a descriptor is found, then that descriptor is retrieved at Step 1637 and if a constant is found, that constant is retrieved at Step 1638. The descriptor or constant can then be immediately used at Step 1639.

If at Steps 1635 and 1636, neither a constant nor a descriptor is found, then at Step 1640, the second level access control registers are accessed and the corresponding access control bits retrieved at Step 1641 using the page index bits from virtual address. At Step 1642, the page index bits from the virtual address are transformed into physical address bits by shifting bit positions. These physical address bits, along with the retrieved access control bits are merged at Step 1643 to form a synthesized descriptor. The synthesized descriptor is returned at Step 1644 for updating the TLB, performing a memory upon a cache miss, or similar operation.

It should be noted that for brevity that the synthesized table walk has been described in terms of only Level 1 and Level 2 descriptor generation. It should be noted however, that additional levels of walk through below the second level can be implemented by repeated application of the inventive principles.

In sum, according to the inventive concepts, only a 32-bit AP register and a pair of 16-bit registers for bufferability and cacheability are needed as first level tables. A second level table composed of a small page AP register and one-bit cacheability and bufferability registers is all that is required for each second level page that must be addressed.

The inventive concepts also advantageously allow for address translation and TLB update upon a cache miss through register emulation of memory similar to the cache miss emulation. Subsequently, the cache and/or TLB entries can be locked as described above for security. The preferred emulation process employs an alternate, emulated memory, such that the integrated circuit 100 internal memory can be spared for other tasks. The memory addresses of the page

tables are preferably mapped inside the integrated circuit.
A preferred procedure embodying these concepts is the
Emulated Table Walk / TLB Update procedure 1700 illustrated
in FIGURE 17.

5   First, an emulated Level 1 Translation Register (table)
(EL1TR) containing either Level 1 Descriptors or Level 2
base addresses is created at (Step 1701). Additionally, an
emulated Level 1 Index Register (EL1IR), maintaining indices
to the entries in the EL1TR, is setup in the alternate
10   memory space (Step 1702). The Translation Base Table (TTB)
in the MMU is programmed to point to the emulated Level 1
table. Requests to this region will receive the contents of
EL1TR with the index into the table matches EL1IR. If the
index does not match the value returned will be an entry
15   that causes an exception to occur.

          For those address translations continuing past the
Section level, an emulated Level 2 Translation Register
(EL2TR) containing Level 2 Descriptors is created in
alternate memory (Step 1704), along with an emulated Level 2
20   Index Register holding the corresponding indices (Step
1705).

          At Step 1706, virtual address is generated by prompting
CPU 101 or through the use of an external address generator.
If the cache and TLB have been flushed or cleaned, a
25   cache/TLB miss will occur, and therefore, at Step 1707, the
table walk procedure is invoked using the emulated level 1
table pointed-to by the MMU. Level 1 Table Index bits in
the virtual address are compared with those in EL1IR and the
corresponding Level 1 information returned from EL1TR (Step
30   1708).

          If the information is a descriptor (i.e. no Level 2
translation is required) at Step 1708,then a Level 1 access
is performed (Step 1709) wherein the permissions in the

descriptor are examined (Step 1710). If permission is not granted, then the operation aborts at Step 1711. Otherwise, the physical address is generated from the Section address bits in the Level 1 descriptor, along with the Section Index from the virtual address, at Step 1712. The physical address can then be loaded into the TLB at Step 1713 to await locking and the corresponding data or instruction loaded into the appropriate cache. If it is determined at Step 1714 that the current entry in the TLB is not the last to be loaded, then at Step 1715 the procedure loops back to Step 1706 to initiate the next table walk. Otherwise, the TLB locking procedure is executed at Step 1716.

If the information from EL1TR is found to be a base address to Level 2 at Step 1708, the Level 2 page walk is invoked at Step 1717. The EL2TR registers are accessed (Step 1718) using the base address from EL1TR. The specific register is indexed using the contents of corresponding EL2IR register by comparison against the index bits from the virtual address (Step 1719). The permissions in the returned Level 2 descriptor are examined at Step 1720. If the access is not allowed, the access is aborted at Step 1721, otherwise the physical address is generated at Step 1723 using the physical address bits from the Level 2 descriptor and index bits from the virtual address. The physical address is loaded into the TLB at Step 1723 to await locking.

If the current TLB entry is the last to be loaded at Step 1724, the TLB lock procedure can be invoked at Step 1725, otherwise, at Step 1726 the procedure jumps back to Step 1706 and the table walk for the next TLB entry is initiated.

In some embodiments of integrated circuit 100, a bare CPU may be employed which does not include a Memory

Management Unit (MMU) or hardware cache. For example, CPU core 101 could be based upon an ARM7tdmi processor 102 alone, without cache 103 or MMU 104. When this option is selected, all software must be stored in memory in a flat

5   memory space. However, this may require the use of external memory (e.g. NOR Flash, SRAM, DRAM). As discussed above, the data in an external memory has the significant disadvantage of potentially being accessed or analyzed by an unauthorized end user.

10   In the embodiment of integrated circuit 100, which does not employ either a hardware cache or an MMU, the security code runs in an supervisor mode. In the supervisor mode, access to specific areas of memory and certain registers are subject to a check against supervisor privilege. The

15 security firmware preferably runs from internal memory, such as SRAM. In the supervisory mode, all other software/firmware is interpreted as running in the user mode and is therefore subject to supervisor privilege checking by the secured software.

20   Although the invention has been described with reference to a specific embodiments, these descriptions are not meant to be construed in a limiting sense. Various modifications of the disclosed embodiments, as well as alternative embodiments of the invention will become

25 apparent to persons skilled in the art upon reference to the description of the invention. It should be appreciated by those skilled in the art that the conception and the specific embodiment disclosed may be readily utilized as a basis for modifying or designing other structures for

30 carrying out the same purposes of the present invention. It should also be realized by those skilled in the art that such equivalent constructions do not depart from the spirit

and scope of the invention as set forth in the appended claims.

It is therefore, contemplated that the claims will cover any such modifications or embodiments that fall within the true scope of the invention.

WHAT IS CLAIMED:

1. A system comprising:

a central processing unit operating in response to a set of instructions for processing information;

an interface for providing access to selected circuitry forming a part of said system on a chip by an external device; and

a set of non-volatile programmable security elements for selectively enabling and disabling the operation of said interface to provide a private environment for processing said information.

2. The system of Claim 1 wherein said interface comprises a JTAG port.

3. The system of Claim 1 wherein said interface comprises an in-circuit emulation port.

4. The system of Claim 1 wherein said interface comprises a port allowing said external device to observe an internal state of said integrated circuit.

5. The system of Claim 1 and further comprising boot memory for storing security initialization code, said security initialization code selectively enabled by programming said set of programmable elements.

6. The system of Claim 5 and further comprising boot memory for storing security initialization code, said security initialization code selectively enabled by programming said set of programmable elements.

7.   The system of Claim 1 and further comprising a cache associated with said central processing unit, said cache including a selected number of lockable entries for storing secure information.

8.   The system of Claim 1 and further comprising a translation look aside buffer, with said CPU, said translation look aside buffer including a selected number of lockable entries for storing addresses to secure information in memory

9.   The system of Claim 1 and further comprising on-chip random access memory including a selected amount of memory space for storing address translation tables.

10.   The system of Claim 1 wherein said set of programmable elements comprises a set of fuses.

11.   The system of Claim 1 wherein said set of programmable elements comprise a set of bond options.

12.   The system of Claim 1 wherein said set of programmable elements comprises a set of antifuses.

13.   The system of Claim 1 wherein said set of programmable elements comprises a set of read-only memory cells.

14.   The system of Claim 1 wherein said set of programmable elements comprises a set of write-once memory cells.

15.   The system of Claim 1 wherein said set of programmable elements comprises a set of FLASH memory cells.

1 16. A method for selective secure operation of a system
2 comprising the steps of:
3   disabling debug circuitry forming a part of the system
4 on a chip at power on reset to prevent access by an
5 unauthorized party to security resources;
6   determining whether a security procedure is called for
7 during system initialization from boot memory; and
8   attempting to execute a selected security procedure
9 when a security procedure is called for during system
10 initialization, comprising the steps of:
11   mapping a vector to the selected location in boot
12   memory storing security code calling a selected
13   security procedure;
14   executing the security code in boot memory to
15   determine whether the called security procedure is
16   valid; and
17   operating the system in a secure environment in
18   response to the called security procedure when the
19   called security procedure is valid.

1   17.   The method of Claim 16 and further comprising the step
2   of booting for unsecured operation if a security procedure
3   is not called for in response to said step of determining,
4   comprising the substeps of:
5   selecting one of internal AND external boot memory
6   options;
7   in response to the selection of the internal boot
8   memory option, mapping the vector to a default location in
9   internal boot memory;
10   in response to the selection of the external boot
11   memory option, mapping the vector to a location in external
12   boot memory;
13   enabling the debug circuitry and
14   executing boot code pointed-to by the vector.

1   18.   The method of Claim 17 wherein said substep of mapping
2   the vector in external memory comprises the further substeps
3   of:
4   remapping a chip select signal controlling the external
5   memory to point-to currently executing memory space; and
6   changing a program counter to the vector such that a
7   fetch of an instruction changing the program counter is
8   completed prior to completion of said step of remapping.

19. The method of Claim 18 wherein said vector comprises a
CPU reset vector.

20. The method of Claim 16 wherein execution of the
security code in boot memory determines that the called
security procedure is invalid and said method further
comprises the steps of:
     remapping the vector to the boot memory to a location
storing second selected security code, the second security
code calling a second security procedure;
     executing the second selected security code in boot
memory to determine if the second security procedure is
valid; and
     operating the system on a chip in a secure environment
in response to the second security procedure when the second
security procedure is valid.

21. The method of Claim 16 wherein said step of executing
the security code in boot memory to determine whether the
called security procedure is valid comprises the substep of
searching for the called security procedure in external
memory coupled to the system on a chip.

22. The method of Claim 16 and further comprising the step
of executing default boot code when the called security
procedure is invalid.

23. The method of Claim 16 wherein said step of determining
if a security procedure is called for during system
initialization comprises the substep of reading the state of
a set of programmable elements.

1   24.   The method of Claim 23 wherein said substep of reading
2   is performed by logic gates.

1   25.   The method of Claim 23 wherein said substep of reading
2   is performed by a central processing unit

1   26.   A method of preventing access and observation of
2   encached information comprising the steps of:
3   generating private information to be encached;
4   storing the private information in memory;
5   updating a translation look aside buffer with
6   descriptors to locations in memory containing the private
7   information;
8   forcing a cache miss to a selected location in cache to
9   be loaded with a selected portion of the private
10   information;
11   retrieving the selected portion of the private
12   information from memory using a corresponding descriptor
13   from the translation look aside buffer;
14   loading the retrieved portion of the private
15   information into the selected location in cache; and
16   locking the selected portion of the private information
17   in the selected location in cache.

1   27.   The method of Claim 26 and further comprising the step
2   of locking the descriptor corresponding to the selected
3   portion of the private information in the translation look
4   aside buffer.

1   28.   The method of Claim 26 wherein said selected location
2   in cache is associated with a replacement counter base and
3   said step of locking comprises the substep of resetting the
4   replacement counter base to a value higher than the
5   replacement counter base associated with the selected
6   location in cache.

1   29. The method of Claim 26 wherein said step of updating
2   the translation look aside buffer comprises the substeps of:
3   setting up a translation table including entries for
4   generating the descriptors to memory locations storing the
5   private information;
6   updating a replacement counter to point to a current
7   translation look aside buffer entry to be filled;
8   forcing a miss to the current translation look aside
9   buffer entry;
10   performing a table walk through the translation table
11 to generate a descriptor associated with private information
12 in memory; and
13   loading the descriptor obtained from the table walk in
14 the current translation look aside buffer entry.

1   30. The method of Claim 26 wherein said step of loading the
2   selected portion of the decoded information in cache
3   comprises the step of loading a cache line in instruction
4   cache.

1   31. The method of Claim 26 wherein said step of loading the
2   selected portion of the private information in cache
3   comprises the step of loading a cache line in data cache.

1   32. The method of Claim 26 wherein said step of setting up
2   a translation table comprises the step of setting up an
3   emulated translation table.

1   33. A method of synthesizing translation tables comprising

2   the steps of:

3   setting up at least one register for storing

4   information controlling access to a plurality of memory

5   spaces;

6   generating a virtual address including a pointer to

7   selected information in the at least one register

8   controlling access to a selected one of the memory spaces;

9   accessing said selected information at said pointer

10   from the at least one register; and

11   generating a physical address to the selected one of

12   the memory spaces from the information accessed from the at

13   least one register.

1   34. The method of Claim 33 wherein the selected information

2   comprises access permissions.

1   35. The method of Claim 33 wherein the selected information

2   comprises cacheability and bufferability bits.

1   36. The method Claim 33 wherein the at least one register

2   comprises a first register for storing access permissions

3   associated with each of the memory spaces, a second

4   register for storing a cacheability bit associated with each

5   of the memory spaces and a third register for storing a

6   bufferability bit associated with each of the memory spaces.

1   37.   The method of Claim 33 wherein the selected information
2   accessed from the at least one register comprises a base
3   address to at least one second level register controlling
4   access to a selected part of a selected one of the memory
5   spaces and said step of generating a physical address
6   comprises the substeps of:
7   accessing selected information in the at least one
8   second level register using the base address and a table
9   index from the virtual address; and
10   generating the physical address from the selected
11   information accessed from the at least one second level
12   register and page index bits from the virtual address.

1   38.   The method of Claim 33 wherein said information
2   includes for each of the memory spaces a pair of access
3   permission bits, a bufferability bit and a cacheability bit.

1   39.   A method of performing an emulated translation table
2   walk comprising the steps of:
3   emulating a translation register including a plurality
4   of entries populated with descriptors;
5   emulating an index register storing indices associated
6   with the entries of the emulated translation register;
7   pointing to the emulated translation register with a
8   translation base pointer;
9   generating an address including index bits to the
10   emulated translation register;
11   comparing the index bits from the address with the
12   indices stored in the index register; and
13   selectively accessing a corresponding descriptor in the
14   translation table in response to said step of comparing.

1   40.   The method of Claim 39 wherein said step of generating
2   an address comprises the step of generating a virtual
3   address forcing a miss to an associated cache.

1   41.   The method of Claim 39 wherein the descriptors comprise
2   selected physical address bits and access permissions and
3   said method further comprises the steps of:
4   determining from the permissions from the descriptor
5   selectively accessed from the emulated translation table
6   whether a corresponding access to memory is allowed; and
7   if the access is allowed, generating a physical address
8   to a location in memory using the physical address bits from
9   the accessed descriptor.

1   42.   The method of Claim 39 and further comprising the steps
2   of:
3   populating selected entries of the translation register
4   with second level base addresses;
5   emulating a second level translation register including
6   a plurality of entries populated with second level
7   descriptors;
8   emulating a second level index register populated with
9   indices to corresponding entries in the second level
10   translation register;
11   accessing the second level translation register with a
12   base address from the translation register;
13   comparing index bits from the virtual address with
14   indices in the second level index register; and
15   in response to said step of comparing, selectively
16   accessing a second level descriptor from the corresponding
17   entry in the second level translation table.

1   43.   The method of Claim 42 wherein the second level
2   descriptors comprise selected second level physical address
3   bits and second level access permissions and said method
4   further comprises the steps of:
5   determining from the second level permissions of the
6   descriptor selectively accessed from the emulated second
7   level translation table whether a corresponding access to
8   memory is allowed; and
9   if the access is allowed, generating a physical address
10   to a location in memory using the second level physical
11   address bits from the accessed second level descriptor.

1   51.   A hand-held audio decoder comprising:

2   a central processing unit operating in response to a

3   set of instructions for decoding a stream of encoded digital

4   audio data;

5   memory for storing said set of instructions; and

6   digital to analog converter circuitry for generating

7   audio from said decoded stream of digital audio data.

1   52.   The audio decoder of Claim 51 wherein said central

2   processing unit comprises an advanced risk machine.

1   53.   The audio decoder of Claim 51 wherein said stream of

2   encoded digital data comprises a stream of MPEGx, Layer 3

3   encoded audio data.

1   54.   The audio decoder of Claim 51 wherein said stream of

2   encoded digital data comprises a stream of ACC encoded

3   digital data.

1   55.   The audio decoder of Claim 51 wherein said stream of

2   encoded digital data comprises a stream of MS Audio encoded

3   digital data.

1   56.   The audio decoder of Claim 51 wherein said decoder is

2   capable of operating correctly from one AA battery for a

3   period of at least one hour.

1 57. A method of synthesized address translation comprising
2 the steps of:
3   setting up at least one global register having a
4 plurality of entries each for storing access control bits
5 for a corresponding region of memory each comprising a
6 plurality of locations having common access characteristics;
7 and
8   setting up an individual register for storing a
9 descriptor corresponding to a region of memory having
10 differing access characteristics;
11   generating an address including an index;
12   in response to a first state of the index, accessing
13 said descriptor from the individual register; and
14   in response to a second state of the index, performing
15 the substeps of:
16   accessing the access control bits from a selected
17   one of the global registers pointed-to by said index;
18   and
19   generating a descriptor by merging the access
20   control bits accessed from the selected one of the
21   global registers with selected bits of said address.

1 58. The method of Claim 57 and further comprising the steps
2   of:
3   setting up a constant register for storing a constant;
4 and
5   in response to a third state of the index, accessing a
6 constant from said constant register.

1 59. The method of Claim 58 wherein the constant register
2 comprises hardwired gates.

1   60. The method of Claim 57 wherein the access control bits
2   comprise access permission bits, cacheability bits and
3   bufferability bits.

1   61. The method of Claim 57 wherein said at least one
2   register comprises a first register having a plurality of
3   entries each for storing access permission bits for a
4   corresponding one of the regions, a second register having
5   a plurality of entries each for storing a cacheability bit
6   for a corresponding one of the regions and a third register
7   having a plurality of entries each for storing a
8   bufferability bit for a corresponding one of the regions.

1   62. The method of Claim 57 wherein said descriptor
2   comprises first level descriptor including an second level
3   index, the method further comprising the steps of:
4   setting up at least one second level global register
5   having a plurality of entries each for storing access
6   control bits for a corresponding region of memory comprising
7   a plurality of locations having common access
8   characteristics; and
9   setting up a second level individual register for
10   storing a descriptor corresponding to a region of memory
11   having differing access characteristics;
12   in response to a first state of the second level index,
13   accessing the descriptor from the second level individual
14   register; and
15   in response to a second state of the second level
16   index, performing the substeps of:
17   accessing said access control bits from a selected
18   one of the second level global registers pointed-to by
19   said second level index; and
20   generating a second descriptor by merging the
21   access control bits accessed from the selected one of
22   the second level global registers with selected bits of
23   the address.

1   63.   The method of Claim 62 and further comprising the
2   steps of:
3   setting up a second level constant register for storing
4   a second level constant; and
5   in response to a third state of the second level index,
6   accessing a second level constant from the second level
7   constant register.

1   64. The method of Claim 58 wherein the second constant

2   register comprises hardwired gates.

1   65. The method of Claim 57 wherein the access control bits

2   comprise access permission bits, cacheability bits and

3   bufferability bits.

CIRCUITS, SYSTEMS AND METHODS

FOR INFORMATION PRIVATIZATION

IN PERSONAL ELECTRONIC APPLIANCES

ABSTRACT OF THE DISCLOSURE

5   A system 100 including a central processing unit 101
operates in response to a set of instructions for processing
information. A port 134 provides access to selected
circuitry forming a part of the system by an external
device. A set of non-volatile programmable security
10   elements 133 selectively enable and disable the operation of
the interface to provide a private environment for
processing the information.

15   Table 1

Address	Name	Default	RD/WR	Size	Comments
0x8000.0000	PADR	0	RW	8	Port A data register
0x8000.0001	PBDR	0	RW	8	Port B data
0x8000.0002	-		-	8	Reserved
0x8000.0003	PDDR	0	RW	8	Port D data
0x8000.0040	PADDR	0	RW	8	Port A data direction register
0x8000.0041	PBDDR	0	RW	8	Port B data
0x8000.0042	-		-	8	Reserved
0x8000.0043	PDDDR	0	RW	8	Port D data direction register
0x8000.0080	PEDR	0	RW	3	Port E data
0x8000.00C0	PEDDR	0	RW	3	Port E data
0x8000.0100	SYSCON	0	RW	32	System
0x8000.0140	SYSFLG	0	RD	32	System status flags register 1
0x8000.0180	MEMCFG 1	0	RW	32	Expansion and ROM memory configuration register 1
0x8000.01C0	MEMCFG 2	0	RW	32	Expansion and ROM memory
0x8000.0200	DRFPR	0	RW	8	DRAM refresh period register
0x8000.0240	INSTR1	0	RD	32	Interrupt status register 1

Address	Name	Default	RD/WR	Size	Comments
0x8000.0280	INTMR1	0	RW	32	Interrupt mask register 1
0x8000.02C0	LCDCON	0	RW	32	LCD control
0x8000.0300	TC1D	0	RW	16	Read/Write
0x8000.0340	TC2D	0	RW	16	Read/Write
0x8000.0380	RTCDR	-	RW	32	Realtime clock data register
0x8000.03C0	RTCMR	-	RW	32	Realtime clock match register
0x8000.0400	PMPCON	0	RW	12	PWM pump control register
0x8000.0440	CODR	0	RW	8	CODEC data I/O register
0x8000.0480	UARTDR1	0	RW	8W/11R	UART1 FIFO data register
0x8000.04C0	UBLCR1	0	RW	32	UART 1 bit rate and line
0x8000.0500	SYNCIO	0	RW	32	Synchronous serial I/O data register for master only SSI
0x8000.0540	PALMSW	0	RW	32	Least significant 32-bit word of LCD palette register

73

Address	Name	Default	RD/WR	Size	Comments
0x8000.0580	PALMSW	0	RW	32	Most significant 32-bit word of LCD palette register
0x8000.05C0	STFCLR	-	WR	-	Write to clear all start up reason flags
0x8000.0600	BLEOI	-	WR	-	Write to clear battery low interrupt
0x8000.0640	MCEOI	-	WR	-	Write to clear media changed interrupt
0x8000.0680	TEOI	-	WR	-	Write to clear tick
0x8000.06C0	TC1EOI	-	WR	-	Write to clear TC1 interrupt
0x8000.0700	TC2EOI	-	WR	-	Write to clear TC2 interrupt
0x8000.0740	RTCEOI	-	WR	-	Write to clear RTC match interrupt
0x8000.0780	UMSEOI	-	WR	-	Write to clear UART modem status changed interrupt
0x8000.07C0	COEOI	-	WR	-	Write to clear CODEC sound interrupt

Address	Name	Default	RD/WR	Size	Comments
0x8000.0800	HALT	-	WR	-	Write to enter the Idle State
0x8000.0840	STDBY	-	WR	-	Write to
0x8000.0880 0x8000.0FFF	Reserved				Write will have no effect, read is undefined
0x8000.1000	FBADDR	C	RW	4	LCD frame buffer start address
0x8000.1100	SYSCON2	0	RW	16	System control register 2
0x8000.1140	SYSFLG2	0	RD	16	System status register 2
0x8000.1240	INSTR2	0	RD	24	Interrupt status register 2
0x8000.1280	INTMR2	0	RW	16	Interrupt mask register 2
0x8000.12C0 – 0x80000.147F	Reserved				Write will have no effect, read is undefined
0x8000.1480	UARTDR2	0	RW	8W/11R	UART2 Data Register
0x8000.14C0	UBLCR2	0	RW	32	UART2 bit rate and line control register
0x8000.1500	SS2DR	0	RW	16	Master/slave SSI2 data Register

Address	Name	Default	RD/WR	Size	Comments
0x8000.1600	SRXEOF	-	WR	-	Write to clear RX FIFO overflow flag
0x8000.16C0	SS2POP	-	WR	-	Write to pop SSI2 residual byte into RX FIFO
0x8000.1700	KBDEOI	-	WR	-	Write to clear keyboard interrupt
0x8000.1800	Reserved	-	WR	-	Do not write to this location. A write will cause the processor to go into an unsupported power savings state.
0x8000.1840 - 0x8000.1FFF	Reserved	-			Write will have no effect, read is undefined

Table 2A

Interrupt	Bit in INTMR1 and INTSR1	Name	Comment
FIQ	0	EXTFIQ	External fast interrupt input (NEXTFIQ pin)
FIQ	1	BLINT	Battery low interrupt
FIQ	2	WEINT	Watchdog expired interrupt
FIQ	3	MCINT	Media changed interrupt

Interrupt	Bit in INTMR1 and INTSR1	Name	Comment
IRQ	4	CSINT	Codec sound interrupt
IRQ	5	EINT1	External interrupt input 1 (NEINT1 pin)
IRQ	6	EINT2	External interrupt input 2 (NEINT2 pin)
IRQ	7	EINT3	External interrupt input 3 (EINT3 pin)
IRQ	8	TC1OI	TC1 underflow interrupt
IRQ	9	TC2OI	TC2 undreflow interupt
IRQ	10	RTCMI	RTC compare match interrupt
IRQ	11	TINT	64 Hz tick interrupt
IRQ	12	UTXINT1	Internal UART 1 transmit FIFO empty interrupt
IRQ	13	URXINT1	Internal UART1 receive FIFO full interrupt
IRQ	14	UMSINT	Internal UART1 modem status changed interrupt
IRQ	15	SSEOTI	Synchronous serial interface 1 end of transfer interrupt

Table 2B

Interrupt	Bit in INTMR2 and INTSR2	Name	Comment
IRQ	0	KBDINT	Key press interrupt

Interrupt	Bit in INTMR2 and INTSR2	Name	Comment
IRQ	1	SS2RX	Master/slave SSI 16 bytes received
IRQ	2	SS2TX	Master/slave SSI 16 bytes transmitted
IRQ	12	UTXINT2	UART2 transmit FIFO empty interrupt
IRQ	13	URXINT2	UART2 receive FIFO full interrupt

TABLE 2C

Interrupt	Bit in INTMR3 and INTSR3	Name	Comment
FIQ	0	MCPINT	MCP interface interrupt

TABLE 3

Interrupt Pin	Input State	Operating State Latency	Idle State Latency	Standby State Latency
NEXTFIQ	Not deglitched; must be active for 20 μs to be detected	Worst case latency of 20 μs	Worst case 20 μs; if only single cycle instructions, less that 1 μs	Including PLL/ocs. setting time, approx. 0.25 s when FASTWAKE=0, or approx. 500 μs when FASTWAKE = 1, or = Idle State if in 13 MHz mode with CLENSL set

Interrupt Pin	Input State	Operating State Latency	Idle State Latency	Standby State Latency
NEINT1-2	not deglitched	Worst case latency of 20 μs	As above	As above
EINT3	Not deglitched	Worst case latency of 20 μs	As above	As above
MEDCHG	Deglitched by 16 kHz clock; must be active for at least 80 μs to be detected	Worst case latenncy of 80 μs	Worst case 80 μs; if only single cycle instructions 61 μs	As above (note difference if in 13 MHz mode with CLKENSL set)

TABLE 4

PE[1]	PE[[0]	Boot Block (NCS0)
0	0	32-bit
0	1	8-bit
1	0	16-bit
1	1	Undefined

TABLE 5A

Address Range	Chip Select
0000.0000-0FFF.FFFF	CS7 (Internal only)
1000.0000-1FFF.FFFF	CS6 (Internal only)
2000.0000-2FFF.FFFF	NCS5
3000.0000-3FFF.FFFF	NCS4
4000.0000-4FFF.FFFF	NCS3
5000.0000-5FFF.FFFF	NCS2
6000.0000-6FFF.FFFF	NCS1
7000.0000-7FFF.FFFF	NCS0

TABLE 5B

Address Range	Chip Select
0000.0000-0FFF.FFFF	NCS0
1000.0000-1FFF.FFFF	NCS1
2000.0000-2FFF.FFFF	NCS2
3000.0000-3FFF.FFFF	NCS3
4000.0000-4FFF.FFFF	NCS4
5000.0000-5FFF.FFFF	NCS5
6000.0000-6FFF.FFFF	CS 6 (Internal only)
7000.0000-7FFF.FFFF	CS7 (Internal only)

TABLE 6

DRAM Address Pins	DRAM Column x16 Mode	DRAM Column x32 Mode	DRAM Row x 16§ Mode	DRAM Row x32 Mode	Pin Name
0	A1*	A2	A9	A10	A[27]/DRA[0]
1	A2	A3	A10	A11	A[26]/DRA[1]
2	A3	A4	A11	A12	A[25]/DRA[2]
3	A4	A5	A12	A13	A[24]/DRA[3]
4	A5	A6	A13	A14	A[23]/DRA[4]
5	A6	A7	A14	A15	A[22]/DRA[5]
6	A7	A8	A15	A16	A[21]/DRA[6]
7	A8	A9	A16	A17	A[20]/DRA[7]
8	A18	A19	A17	A18	A[19]/DRA[8]
9	A20	A21	A19	A20	A[18]/DRA[9]
10	A22	A23	A21	A22	A[17]/DRA[10]
11	A24	A25	A23	A24	A[16]/DRA[11]
12	A26	A27	A25	A26	A[15]/DRA[12]

ATTORNEY DOCKET NO.
___-CS

PATENT

TABLE 7

Size	Address Configuration	Total Size of Bank	Address Range of Segments	Size of Segments
4 Mbit	9 Row x 9 Column	1 Mbyte	n000.0000-n00F.FFFF	1 Mbyte
16 Mbit	10 Row x 10 Column	4 Mbytes	n000.0000-n03F.FFFF	4 Mbytes
16 Mbit	12 Row x 8 Column	4 Mbytes	n000.0000-n007.FFFF n010.0000-n017.FFFF n040.0000-n047.FFFF n050.0000-n057.FFFF n100.0000-n107.FFFF n110.0000-n117.FFFF n140.0000-n147.FFFF n150.0000-n157.FFFF	512 KBYTES 512 KBYTES 512 KBYTES 512 KBYTES 512 KBYTES 512 KBYTES 512 KBYTES 512 KBYTES
64 Mbit	11 Row x 11 Column	16 Mbytes	n000.0000-n0FF.FFFF	16 Mbytes

5

Size	Address Configuration	Total Size of Bank	Address Range of Segments	Size of Segments
64 Mbit	13 Row x 9 Column	16 Mbytes	n000.0000-n01F.FFFF n040.0000-n05F.FFFF n100.0000-n11F.FFFF n140.0000-n15F.FFFF n400.0000-n41F.FFFF n440.0000-n45F.FFFF n500.0000-n51F.FFFF n540.0000-n55F.FFFF	2Mbytes 2Mbytes 2Mbytes 2Mbytes 2Mbytes 2Mbytes 2Mbytes 2Mbytes
256 Mbit	12 Row x 12 Column	64 Mbytes	n000.0000-n3FF.FFFF	64 Mbytes
1Gbit	13 Row x 13 Column	256 Mbytes	n000.0000-nFFF.FFFF	256 MBytes

TABLE 8

Size	Address Configuration	Total Size of Bank	Address Range of segments(s)	Size of segment(s)
4 Mbit	9 Row x 9 Column	0.5 Mbyte	n000.0000-n007.FFFF	0.5 MByte
16 Mbit	10 Row x 10 Column	2 Mbytes	n000.0000-n01F.FFFF	2 Mbytes

PATENT

Size	Address Configuration	Total Size of Bank	Address Range of Segments(s)	Size of Segment(s)
16 Mbit	12 Row x 8 Column	2 Mbytes	n000.0000-n003.FFFF n008.0000-n00B.FFFF n020.0000-n023.FFFF n028.0000-n02B.FFFF n080.0000-n083.FFFF n088.0000-n08B.FFFF n0A0.0000-n0A3.FFFF n0A8.0000-n0AB.FFFF	256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes
64 Mbit	11 Row x 11 Column	8 Mbytes	n000.000-n07F.FFFF	8 Mbytes
64 Mbit	13 Row x 9 Column	8 Mbytes	n000.0000-n00F.FFFF n020.0000-n02F.FFFF n080.0000-n08F.FFFF n0A0.0000-n0AF.FFFF n200.0000-n20F.FFFF n220.0000-n22F.FFFF n280.0000-n28F.FFFF n2A0.0000-n2AF.FFFF	1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte
256 Mbit	12 Row x 12 Column	32 Mbytes	n000.0000-n1FF.FFFF	32 Mbytes
1 Gbit	13 Row x 13 Column	128 Mbytes	n000.0000-n7FF.FFFF	128 Mbytes

TABLE 9

Address (W/B)	Operating	Idle	Standby	NPOR RESET	NRESET RESET
DRAM Control	On	On	SELFREF	Off	SELFREF
UARTs	On	On	Off	Reset	Reset
LCD FIFO	On	On	Reset	Reset	Reset
LCD	On	On	Off	Reset	Reset
ADC Interface	On	On	Off	Reset	Reset
SS12 Interface	On	On	Off	Reset	Reset
DAI Interface	On	On	Off	Reset	Reset
Codec	On	On	Off	Reset	Reset
Timers	On	On	Off	Reset	Reset
RTC	On	On	On	On	On
LED Flasher	On	On	On	Reset	Reset
DC-to-DC	On	On	Off	Reset	Reset
CPU	On	Off	Off	Reset	Reset
Interrupt Control	On	On	On	Reset	Reset
PLL/CLKEN Signal	On	On	Off	Off	Off

ATTORNEY DOCKET NO.
_____-CS

TABLE 10

Pin No. LQFP	External Pin Name	SSI2 Slave Mode (Internal Name)	SSI2 Master Mode	Codec Internal Name	DAI	Strength
63	SSICCLK	SSICCLK= serial bit clock; Input	Output	PCMCLK = Output	SCLK = Output	1
65	SSITXFR	SSKTXFR=T X frame sync; Input	Output	PCMSYNC = Output	LRCK = Output	1
66	SSITXDA	SSITXDA=T X data; Output	Output	PCMOUT = Output	SDOUT = Output	1
67	SSIRXDA	SSIRXDA=R X data; Input	Input	PCMIN = Input	SDIN = Input	
68	SSIRXFR	SSIRXFR=R X frame sync; Input	Output	p/u*	MCLK	1

*p/u = use an 10 k pull-up

TABLE 11

Type	Comments	Referred To As	Max. Transfer Speed
SP/Microwire 1	Master mode only	ADC Interface	128 Kbps
SPI/Microwire 2	Master/slave mode	SSI2 Interface	512 Kbps
MCP Interface	CD quality DACs and ADCs	DAI	1.536 Mbps
Codec Interface	**Only** for use in the PLL clock mode	Codec Interface	64 Kpbs

TABLE 12

SYSCON1 Bit 17	SYSCON1 Bit 16	13.0 MHz Operation ADCCLK Frequency (kHz)	18. 432-73.728 MHz Operation ADCCLK Frequency (kHz)
0	0	4.2	4
0	1	16.9	16
1	0	67.7	64
1	1	135.4	128

ATTORNEY DOCKET NO.
____-CS

TABLE 13

Address (W/B)	Data In Memory	Byte Lanes to Memory/Ports/Registers								RO Contents	
		Big Endian Memory				Little Endian Memory				Big Endian	Little Endian
		7:0	15:8	23:16	31:24	7:0	15:8	23:16	31:24		
Word +0 (W)	11223344	44	33	22	11	44	33	22	11	11223344	11223344
Word +1 (W)	11223344	44	33	22	11	44	33	22	11	44112233	44112233
Word +2 (W)	11223344	44	33	22	11	44	33	22	11	33441122	33441122
Word + 3 (W)	11223344	44	33	22	11	44	33	22	11	22334411	22334411
Word +0 (B)	11223344	dc	dc	dc	11	44	dc	dc	dc	00000001	00000004
Word +1 (B)	11223344	dc	dc	22	dc	dc	33	dc	dc	00000002	00000003
Word +2 (B)	11223344	dc	33	dc	dc	dc	dc	22	dc	00000003	00000002
Word +3 (B)	11223344	44	dc	dc	dc	dc	dc	dc	11	00000004	00000001

TABLE 14

Address (W/B)	Register Contents	Byte Lanes to Memory/Ports/Registers							
		Big Endian Memory				Little Endian Memory			
		7:0	15:8	23:16	31:24	7:0	15:8	23:16	31:24
Word +0 (W)	11223344	44	33	22	11	44	33	22	11
Word +1 (W)	11223344	44	33	22	11	44	33	22	11
Word +2 (W)	11223344	44	33	22	11	44	33	22	11
Word +3 (W)	11223344	44	33	22	11	44	33	22	11
Word +0 (B)	1223344	44	44	44	44	44	44	44	44
Word +1 (B)	11223344	44	44	44	44	44	44	44	44
Word +2 (B)	11223344	44	44	44	44	44	44	44	44
Word +3 (B)	11223344	44	44	44	44	44	44	44	44

TABLE 15

Name	Bits	Address	Comment
UNIQID	31:0	0x2440	32 bit ID
UNIQCHK	39:32	0x2450	8 bit hamming code for UNIQID
UNIQID2	159:128	0x2700	32 bit ID #2
UNIQID3	191:160	0x2704	32 bit ID #3
UNIQID4	223:192	0x2708	32 bit ID #4
UNIQID5	255:224	0x270C	Hamming codes and flags
UNIQCHK2	231:224		8 bit hamming code for UNIQID2
UNIQCHK3	239:232		8 bit hamming code for UNIQID3
UNIQCHK4	247:240		8 bit hamming code for UNIQID4
ASECEX	248		Alternate security exists
	255:249		Reserved `0'

TABLE 16

Name	Comment
PRIVID	PRIVATE ID number
PRIVHAM	PRIVATE hamming code for private ID numbers
PRIVFLG	Private firmware exists

TABLE 17

Bit	Definition
0	Good Validation
1	ID all 0's
2	CHK all 0's
3	ID all 1's
4	CHK all 1's

TABLE 18

Name	Address	ID-CHK pair
UNIQVAL	0x2460	UNIQID-UNIQCHK
UNIQVAL2	0x2720	UNIQID2-UNIQCHK2
UNIQVAL3	0x2724	UNIQID3-UNIQCHK3
UNIQVAL4	0x2728	UNIQID4-UNIQCHK4

TABLE 19

Name	Address	ID-CHK pair
SECVAL1	0x2540	SECID1-SECCHK1
SECVAL2	0x2544	SECID2-SECCHK2

TABLE 20

Name	Address	Comment
TESTID	0x27AC	32 bit general register
TESTCHK	0x2753	8 bit general register
TESTVAL	0x2744	Validation for TESTID-TESTCHK pair

Fig. 1A

Block diagram with the following labeled elements:

100

132 — D0-D31

INTERNAL DATA BUS

121 FLASH INTERFACE — FLASH CARD, FLASH MEM
106

119 EXPANSION CONTROL — EXPCLK, WORD, CS[0-3], EXPRDY, WRITE

120 DRAM CONTROLLER — PARALLEL PORT, EPP
104

133 INTERNAL ADDRESS BUS — A[0-27],

103

123 LCD DMA

134 ICE-JTAG — TEST AND DEVELOPEMENT

107 LCD CONTROLLER — LCD DRIVE
122

124 SRAM 37.5 KBytes
118

131 APB BRIDGE

UART1 — IrDA — LED AND PHOTODIODE
126

ASYNC INTERFACE 1

UART2 — ASYNC INTERFACE 2
125

101

ARM720T
102
- OARM7TDMI CPU CORE
- 8-KBYTE CACHE
- MMU
- WRITE BUFFER
105

TIMER COUNTER (2)
113

ON-CHIP BOOT ROM
117

APB BUS
132

SECURITY REGISTERS & HARDWARE
133

108

109 PLL — 3.6864 MHz
111

32,768-KHz OSCILLATOR — 32.768 KHz
110

STATE CONTROL — POR, RUN, RESET, WAKEUP
114

POWER MANAGEMENT — BATOK, EXTPWR PWRFL, BATCHG
115

INTERRUPT CONTROLLER — EINT[1-3], FIQ, MEDCHG
116

RTC

FLASHING LED DRIVE
112

GPIO — PORTS A, B, D (8-BIT) PORT E (3-BIT)
129

PWM — DC TO DC
130

SSI (ADC) — ADCCLK, ADCIN, ADCOUT, SMPCLK, ADCCS
127

Serial interface — SISCLK, SSITXFR, SSITXDA, SSIRXDA, SSIRSFR
128

155	154	156	157
JTAG I/F	8 CHANNEL DMA ENGINE w/ CRC GENERATOR	USB HOST CONTROLLER w/ 3 PORTS	10/100Mb/s ETHERNET

AHB

150 TIC I/F 151

PCMCIA I/F

EXTERNAL AMBA I/F 142

SDRAM I/F 149

SRAM/FLASH/ROM I/F 152

AHB/APB BRIDGE 144

BRIDGE 145

A H B

ARM920T w/ 16KB I-CACHE, 16KB D-CACHE, MMU 141

BOOT ROM 153

RASTER ENGINE SUPPORTING B&W, COLOR LCD's, NTSC/ PAL ENCODERS 147

148

SECURITY REGISTERS 133

GRAPHICS ENGINE 146

UART w/ MODEM CTRL. 158

UART w/ 115b/s & 4Mb/s IrDA 159

UART w/ HDLC 160

RTC w/TRIM 165

MMC I/F 164

A P B 143

EEPROM/ 12C I/F 168

AC97 I/F 163

SPI I/F 161

SPI I/F 162

WATCHDOG TIMER 167

16 BIT TIMERS (4), 32 BIT TIMER, 32 BIT DEBUG TIMER 166

INTERRUPT CTRL. 174

GPIO (16) 173

LED OUTPUTS (2) 172

PLLs 175

SYSTEM CONTROL 176

REMAP & PAUSE 177

FLASH VPP CTRL. 178

8 WIRE TOUCHSCREEN I/F 170

12 BIT TOUCHSCREEN ADC 171

6 BIT CONTRAST DAC

8x8 KEY MATRIX I/F 169

APB

Fig. 1B

FIG. 1C

FIG. 2

200

100

CRYSTAL — 109

MOSCIN

CS[4]
PB0
EXPCLK

PC CARD SOCKET — 205

CL-PS6700 PC CARD CONTROLLER — 206

D[31:0]

A[27:0]

MOE
WRITE

207

×16 DRAM   ×16 DRAM

×16 DRAM   ×16 DRAM

RAS[1]
RAS[0]
CAS[0]
CAS[1]

CAS[2]
CAS[3]

CS[0]
CS[1]

208

×16 FLASH   ×16 ROM

×16 FLASH   ×16 ROM

CS[n]
WORD

204

EXTERNAL MEMORY-MAPPED EXPANSION   BUFFERS

ADDITIONAL I/O — BUFFERS AND LATCHES

CS[2]
CS[3]

DD[3:0]

CL1
CL2
FM
M

COL[7:0]

PA[7:0]

PB[7:0]

PD[7:0]

PE[2:0]

POR
PWRFL
BATOK
EXTPWR
BATCHG
RUN
WAKEUP

DRIVE[1:0]

FB[1:0]

SSICLK
SSITXFR
SSITXDA
SSIRXDA

LEDDRV
PHDIN

RXD1/2
TXD1/2
DSR
CTS
DCD

ADCCLK
ADCCS
ADCOUT
ADCIN
SMPCLK

LCD MODULE — 209

KEYBOARD — 215

DC INPUT — 202

POWER SUPPLY UNIT AND COMPARATORS
201

BATTERY — 203

DC-TO-DC CONVERTERS — 216

CODEC — 214

IR LED AND PHOTODIODE — 211

2× RS-232 TRANSCEIVERS — 210

ADC — 212   DIGITIZER — 213

FIG. 3

- 142
- BUS I/F 305
- WRITE BUFFER 306
- WRITE BACK PATAG RAM 307
- 32-bit WBPA
- INSTRUCTION CACHE 301
- DATA CACHE 302
- INSTRUCTION MMU 303
- DATA MMU 304
- 32-bit IPA
- 32-BIT ID
- 32-BIT DD
- 32-BIT DPA
- 32-bit IMVA
- 32-BIT DVMA
- PROCESSOR CORE AND COPROCESSOR 141

13 MHz

CLKEN

Fig. 4A

EXPCLK

(int) RUN

CLKEN

intr./
WAKEUP

T42

Fig. 4B

OPERATING

WRITE TO HALT LOCATION

INTERRUPT OR RISING WAKEUP

WRITE TO STANDBY LOCATION,
POWER FAIL, OR USER RESET

INTERRUPT,RISING WAKEUP

IDLE

STANDBY

NPOR, POWER FAIL, OR
USER RESET

Fig. 5

FIG. 6

FIG. 7A

ADCCLK 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 22 23

NADCCS

ADCIN DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0

ADCOUT DO15 DO14 DO13 DO12 DO11 DO10 DO1 DO0

SSI Timing Interface for MAX148/9

FIG. 7b

SSI Interface for AD7811/2

ADCCLK

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 22 23

NADCCS

ADCIN   DI9 DI8 DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0

ADCOUT   DO9 DO8 DO1 DO0

FIG. 8

CDENRX

CDENTX

CSINT

|← 1 ms →| |← 1 ms →| |← 1 ms →|

INTERRUPT OCCURS

INTERRUPT OCCURS

INTERRUPT OCCURS

900

F/G. 9

DAI ADC

| SCLK |
| LRCK |
| SDATA |
| MCLK |

~901

DAI DAC

| SCLK |
| LRCK |
| SDATA |
| MCLK |

~902

CLOCK OPEN

~903

SDIN
SCLK
LRCK
SDOUT
MCLK

System
100

12S INTERFACE

FIG. 10

128 SCLKs

Left Channel | Right Channel

LRCK

SCLK

SDATAO msb | -1 | -2 | -3 | -4 | -5 | +5 | +4 | +3 | +2 | +1 | lsb

SDATAI msb | -1 | -2 | -3 | -4 | -5 | +5 | +4 | +3 | +2 | +1 | lsb

FIG. 11

MASTER		SLAVE
SSIRXFR	→	SSIRXFR
SSITXFR	→	SSITXFR
SSICLK	→	SSICLK
SSITXDA	→	SSIRXDA
SSIRXDA	←	SSITXDA

FIG. 12
(SHEET 1),

OPTIONAL PROCEDURE ~ 1200

POWER ON RESET ~ 1201

DISABLE ALL HARDWARE DEBUG FEATURES AND HIDE ALL SECURITY ELEMENTS ~ 1202

SECURITY EXISTS ? ~ 1203

Yes → EXECUTE SECURITY PROCEDURE (SHEET 3) ~ 1208

NO → SHEET 2

EXECUTE "NORMAL BOOT" (FROM SHEET 4) ~ 1219

ENABLE ALL DEBUG FEATURES AND HIDE ALL SECURITY FEATURES ~ 1220

SELECT DEFAULT BOOT ROM ~ 1221

DYNAMICALLY FORCE PROCESSOR TO RESET VECTOR ~ 1222

FIG 12
sheet 2

FROM SHEET 1

BOOT FROM INTERNAL MEMORY ? ~1204

YES

NO

RESET VECTOR MAPS TO DEFAULT INTERNAL BOOT LOCATION ~1205

RESET VECTOR MAPS TO EXTERNAL MEMORY ~1206

PROCESSOR EXECUTES FROM RESET VECTOR ~1207

OUTGOING ELEMENTS

FIG. 12
sheet 3

EXECUTE SECURITY
PROCEDURE
(FROM SHEET 1) ~1208

DETERMINE FIRST
SECURITY IMPLEMENTATION
AND SELECT ~1209

MAP RESET VECTOR TO
SELECTED BOOT LOCATION ~1210

EXECUTE BOOT CODE FOR
NEXT SECURITY IMPLEMENTATION
(FROM SHEET 5) ~1218

ENABLE SECURITY ELEMENTS OF
SELECTED SECURITY
IMPLEMENTATION AND HIDE ALL
OTHER SECURITY
IMPLEMENTATIONS ~1211

SHEET
4

FIG.12
Sheet 4

```
┌──────────┐
│   FROM   │
│  SHEET   │
│    3     │
└──────────┘
     │
     ▼
┌────────────────────┐
│ EXECUTE SELECTED   │──~ 1212
│ BOOT CODE WITH     │
│ PROCESSOR TO       │
│ INITIALIZE FOR     │
│ SELECTED SECURITY  │
│ IMPLEMENTATION     │
└────────────────────┘
     │
     ▼
      ~ 1213
    ◇ SYSTEM VALID ◇ ──YES──→ ┌──────────────────────┐
    ◇ FOR SECURITY ◇          │ SYSTEM RUNS IN       │──~ 1214
    ◇ IMPLEMENTATION◇         │ SECURE ENVIRONMENT   │
    ◇     ?        ◇          │ FROM SELECTED        │
      │ NO                    │ SECURITY IMPLEMENTATION│
      ▼                       └──────────────────────┘
      ~ 1215
    ◇ LAST SECURITY ◇──YES──→ ┌──────────────────────┐
    ◇ IMPLEMENTATION◇         │ EXECUTE NORMAL       │──~ 1219
    ◇     ?        ◇          │ BOOT                 │
      │ NO                    │ (SHEET 1)            │
      ▼                       └──────────────────────┘
┌──────────┐
│  SHEET   │
│    5     │
└──────────┘
```

FIG. 12
Sheet 5

FROM
SHEET
4

DETERMINE NEXT
SECURITY
IMPLEMENTATION AND
SELECT

~ 1216

DYNAMICALLY FORCE
PROCESSOR TO RESET
VECTOR

~ 1217

EXECUTE BOOT CODE
FOR NEXT SECURITY
IMPLEMENTATION
(SHEET 3)

~ 1218

FIG. 13

(Sheet 1)

```
        ┌─────────────────┐
        │  TLB LOCKDOWN   │ ～1300
        └────────┬────────┘
                 │
                 ▼
    ┌─────────────────────┐
    │  SET-UP ACTUAL OR   │ ～1301
    │  EMULATED PAGE      │
    │       TABLE         │
    └──────────┬──────────┘
               │
               ▼
    ┌─────────────────────┐
    │  FLUSH AT LEAST     │ ～1302
    │  SOME TLB ENTRIES   │
    │  TO INSURE CODE TO  │
    │  BE LOCKED NOT      │
    │  ALREADY            │
    │     IN TLB          │
    └──────────┬──────────┘
               │
               ▼
    ┌─────────────────────┐      ┌──────────────────┐
    │  UPDATE             │ ～1303│ FILL NEXT ENTRY  │ ～1309
    │  REPLACEMENT        │◄─────┤                  │
    │  COUNTER TO         │      └──────────────────┘
    │  NEXT ENTRY IN      │
    │       TLB           │
    └──────────┬──────────┘
               │
               ▼
          ┌─────────┐
          │ SHEET   │
          │   2     │
          └─────────┘
```

FIG. 13

(Sheet 2)

FROM
SHEET
1

EXECUTE LOAD OR
PREFETCH INSTRUCTION
TO FORCE MISS TO DATA ∿ 1304
TLB OR INSTRUCTION
TLB

PEFORM PAGE TABLE ∿ 1305
WALK TO OBTAIN
DESCRIPTOR

LOAD
DESCRIPTOR ∿ 1306
INTO TLB ENTRY AT
REPLACEMENT
COUNTER

SHEET
3

FIG. 13
(Sheet 3)

```
FROM
SHEET
2
```

LOCKDOWN TLB
ENTRY BY SETTING
BIT IN TLB
LOCKDOWN REGISTER ~ 1307

~ 1308

LAST ENTRY
? — NO → FILL NEXT ENTRY
(SHEET 1) ~ 1309

YES

END

FIG. 14
(Sheet 1)

CACHE LOCKDOWN ~ *1400*

SET-UP ACTUAL OR
EMULATED PAGE
AND UPDATE TLB ~ *1401*

FLUSH AT LEAST
SOME CACHE LINES
TO INSURE CODE TO
BE LOCKED NOT
ALREADY
ENCACHED ~ *1402*

LOAD NEXT CACHE LINE

UPDATE
REPLACEMENT
COUNTER TO
NEXT CACHE LINE IN
CACHE SEGMENT
INDEXED BY
VIRTUAL ADDRESS ~ *1403*

SHEET
2

FIG. 14
(Sheet 2)

FROM
SHEET
1

DECODE AND STORE
IN MEMORY CODE OR
DATA TO BE LOCKED
IN CACHE ~ 1404

FORCE CACHE MISS
AND GENERATE
PHYSICAL ADDRESS
FROM DATA IN
CORRESPONDING
TLB ENTRY ~ 1405

PERFORM LINE FILL
AT REPLACEMENT
COUNTER POINTER
AND INDEXED
CACHE SEGMENT ~ 1406

~ 1407

LAST SEGMENT
?

1408 ~ UPDATE SEGMENT
INDEX IN VIRTUAL
ADDRESS

NO

YES

SHEET
3

FIG. 14
(Sheet 3

FROM
SHEET
2

LAST CACHE
LINE
?

~ 1409

NO

Yes

LOAD NEXT CACHE LINE ~ 1410

SET REPLACEMENT
COUNTER BASE
TO ONE HIGHER
THAN BASE TO
LOCKED CACHE LINES

~ 1411

EXECUTE CODE
LOCKED IN CACHE

~ 1412

EMULATED
CACHE MISS ~ 1500

WRITE INFORMATION TO BE
ENCACHED INTO ECLINE
REGISTERS ~ 1501

WRITE CORRESPONDING
OFFSET TO LOCATION
IN LOCKABLE
CACHE SPACE IN
ECOFFSET REGISTER ~ 1503

PERFORM PREFETCH OR
DATA LOAD OPERATION
TO CACHE LOCATION
TO BE LOCKED AND
FORCE CACHE MISS ~ 1503

SHEET
2

FIG. 15
(sheet 2)

FROM
SHEET
1

RETRIEVE INFORMATION
FROM CORRESPONDING
ECLINE REGISTERS
USING OFFSET
COMPARISON WITH
CACHE PHYSICAL
ADDRESS   ~1504

LOAD INFORMATION
FROM ECLINE
REGISTERS INTO
CACHE LOCATION
TO BE LOCKED   ~1505

CONTINUE CACHE LOCK
PROCEDURE

UPDATE SYNTHESIZED TABLES FOR NEXT MEMORY BLOCK ~ 1600

COMMON ACCESS CHARACTERISTICS ? ~ 1601

NO

YES

LOAD ENTRY IN REGISTER STORING GLOBAL LEVEL 1 AP BITS ~ 1602

LOAD ENTRY IN REGISTER STORING LEVEL 1 GLOBAL BUFFERABILITY BITS ~ 1603

LOAD ENTRY IN REGISTER STORING LEVEL 1 GLOBAL BUFFERABILITY BITS ~ 1604

UPDATE SYNTHESIZED TABLES FOR NEXT MEMORY BLOCK ~ 1605

FULL REGISTER FOR DESCRIPTOR ? ~ 1606

YES

NO

LOAD FULL REGISTER WITH COMPLETE LEVEL 1 DESCRIPTOR ~ 1607

UDATE SYTHESIZED TABLES FOR NEXT MEMORY BLOCK ~ 1608

STORING CONSTANT IN HARDWIRED GATES ~ 1609

LEVEL 2 BASE ADDRESS ~ 1610

UPDATE SYNTHESIZED TABLES FOR NEXT MEMORY BLOCK ~ 1611

UPDATE LEVEL 2 SYNTHESIZED TABLE (SHEET 2)

FIG. 16 A
(Sheet 2)

FROM
SHEET 1

COMMON ACCESS
CHARACTERISTICS
? ~1612

LOAD ENTRY IN
REGISTER
STORING GLOBAL
LEVEL 2 AP BITS ~1613

LOAD ENTRY IN
REGISTER
STORING GLOBAL
LEVEL 2
BUFFERABILITY BITS ~1614

LOAD ENTRY
IN REGISTER
STORING GLOBAL
LEVEL 2
CACHEABILITY BITS ~1615

UPDATE SYNTHESIZED
TABLE FOR NEXT
MEMORY BLOCK ~1616

FULL REGISTER
FOR
DESCRIPTOR ~1612a

LOAD FULL REGISTER
WITH COMPLETE
LEVEL 2
DESCRIPTOR ~1617

STORE CONSTANT
IN HARDWIRED
GATES ~1618

UPDATE SYNTHESIZED
TABLES FOR NEXT
MEMORY BLOCK ~1619

UPDATE SYNTHESIZED
TABLES FOR NEXT
MEMORY BLOCK ~1619

REQUEST FOR DESCRIPTOR DURING TABLE WALK ～1620

LEVEL 2 TABLE ? ～1621

TO SHEET 3

POINT TO LEVEL 1 REGISTERS FROM MMU TRANSLATION TABLE BASE REGISTER ～1622

INDEX LEVEL 1 REGISTERS USING TABLE INDEX BITS FROM VIRTUAL ADDRESS ～1623

FULL REGISTER ? 1624

RETURN FULL LEVEL 1 DESCRIPTOR FROM REGISTER 1630

CONSTANT ? 1625

RETURN CONSTANT FROM HARDWIRED GATES 1631

ACCESS INDEXED BITS IN GLOBAL ACCESS CONTROL REGISTERS ～1626

USE RETURNED INFORMATION ～1632

TO SHEET 2

```
  FROM
  SHEET
    1
```

TRANSFORM
TABLE INDEX
FROM VIRTUAL
ADDRESS TO
PHYSICAL ADDRESS
BITS
~ 1627

MERGE ACCESS
CONTROL BITS
WITH PHYSICAL
ADDRESS BITS
TO FORM
DESCRIPTOR
~ 1628

RETURN
SYNTHESIZED
DESCRIPTOR
~ 1629

```
  FROM
  SHEET
    3
```

RETRIEVE ACCESS
CONTROL BITS
AT PAGE INDEX
~ 1641

TRANSFORM
PAGE INDEX BITS
FROM VIRTUAL
ADDRESS TO
PHYSICAL ADDRESS
BITS
~ 1642

MERGE ACCESS CONTROL
BITS AND PHYSICAL
ADDRESS BITS TO
FORM DESCRIPTOR
~ 1643

RETURN SYNTHESIZED
DESCRIPTOR
~ 1644

FROM
SHEET
1

POINT TO LEVEL 2
REGISTERS FROM MMU
MEMORY ADDRESS ~ 1633

INDEX LEVEL 2
REGISTERS USING
TABLE INDEX BITS
FROM VIRTUAL
ADDRESS ~ 1634

FULL
REGISTER
? ~ 1635

RETURN FULL LEVEL 1
DESCRIPTOR FROM
REGISTER 1637

CONSTANT
? ~ 1636

RETURN CONSTANT
FROM HARDWIRED
GATES 1638

ACCESS INDEXED
BITS IN
SECOND LEVEL
ACCESS CONTROL
REGISTERS ~ 1640

USE RETURNED
INFORMATION ~ 1639

TO
SHEET
2

FIG. 17

Sheet 1

EMULATED TABLE WALK ~ 1700

CREATE EMULATED LEVEL 1 TRANSLATION REGISTER CONTAINING LEVEL 1 DESCRIPTORS OR LEVEL 2 BASE ADDRESSES ~ 1701

CREATE EMULATED LEVEL 1 INDEX REGISTER TO EMULATED LEVEL 1 TRANSLATION REGISTER ~ 1702

PROGRAM LEVEL 1 TRANSLATION BASE POINTER IN MMU TO POINT TO EMULATED LEVEL 1 TRANSLATION REGISTER ~ 1703

CREATE EMULATED LEVEL 2 TRANSLATION TABLE CONTAINING LEVEL 2 DESCRIPTORS ~ 1704

SHEET 2

FIG. 17
sheet 2

FROM
SHEET
1

CREATE EMULATED
LEVEL 2 INDEX
REGISTER TO LEVEL 2
EMULATED LEVEL 2
TRANSLATION REGISTER ~1705

INITIATE TABLE WALK
FOR NEXT ENTRY ~1715 / 1726

GENERATE VIRTUAL ADDRESS
AND
FORCE CACHE MISS TO
INITIATE EMULATED
TABLE WALK ~1706

COMPARE VIRTUAL ADDRESS
LOOKUP TABLE INDEX BITS
WITH BITS IN EMULATED
LEVEL 1 INDEX REGISTER
TO ACCESS LEVEL 1
TRANSLATION REGISTER ~1707

LEVEL 2
DESCRIPTOR
? ~1708

PERFORM LEVEL 2
LOOKUP
(SHEET 4) ~1717

PERFORM LEVEL 1
ACCESS
(SHEET 3) ~1709

F1G.17
sheet 3

FROM
SHEET 2

1710

ACCESS
PERMISSION
?

NO

ABORT ~1711

YES

GENERATE PHYSICAL
ADDRESS USE
BASE ADDRESS FROM
EMULATED LEVEL 1
TRANSLATION
REGISTER AND SECTION
INDEX FROM VIRTUAL
ADDRESS ~1712

LOAD
PHYSICAL
ADDRESS IN TLB ~1713

INTIATE TABLE WALK
FOR NEXT ENTRY
(SHEET 2) ~1715

LAST
ENTRY
? ~1714

EXECUTE TLB
LOCK PROCDURE ~1716

FIG. 17

Sheet 4

```
┌──────────┐
│   FROM   │
│  SHEET   │
│    2     │
└──────────┘
     │
     ▼
┌─────────────────────┐
│  ACCESS LEVEL 2     │  ~1718
│  EMULATED           │
│  TRANSLATION REGISTER│
│  WITH BASE ADDRESS FROM│
│  LEVEL 1 REGISTER   │
└─────────────────────┘
     │
     ▼
┌─────────────────────┐
│  COMPARE LEVEL 2    │  ~1719
│  INDEX FROM VIRTUAL │
│  ADDRESS WITH LEVEL 2│
│  INDEX REGISTER TO ACCESS│
│  LOCATION IN LEVEL 2│
│  TRANSLATION REGISTER│
└─────────────────────┘
     │
     ▼
    ◇◇◇
  ◇     ◇   ~1720
 ◇ ACCESS ◇
 ◇PERMISSION◇──────►┌──────────┐
 ◇    ?   ◇          │  ABORT   │ ~1721
  ◇     ◇            └──────────┘
    ◇◇◇
     │
     ▼
┌──────────┐
│  SHEET   │
│    5     │
└──────────┘
```

FROM
SHEET
4

GENERATE PHYSICAL
ADDRESS FROM BASE
ADDRESS FROM LEVEL 2
TRANSLATION REGISTER
AND PAGE INDEX
FROM VIRTUAL
ADDRESS — 1722

LOAD PHYSICAL
ADDRESS IN TLB — 1723

LAST TLB
ENTRY
? — 1724

INITIATE TABLE WALK
FOR NEXT ENTRY
(SHEET 2) — 1726

EXECUTE TLB
LOCK PROCEDURE — 1725

DECLARATION AND POWER OF ATTORNEY
Original Application

As below named inventor, I declare that I have reviewed and understand the contents of the specification, including the claims, as amended by any amendment specifically referred to in this Declaration, that the information given herein is true, that I believe that I am the original, first and joint inventor of the invention entitled:

CIRCUITS, SYSTEMS AND METHODS FOR INFORMATION PRIVATIZATION IN PERSONAL ELECTRONIC APPLIANCES

which is described and claimed in:

__XX__   the attached specification or

___   the specification in application Serial No. _____ filed _____

that I acknowledge my duty to disclose information in accordance with 37 C.F.R. Section 1.56 and defined on the attached sheet, which is material to the examination of this application, that I do not know and do not believe the same was ever known or used in the United States of America before my or our invention thereof or patented or described in any printed publication in any country before my or our invention thereof, or more than one year prior to this application, that the invention has not been patented or made the subject of an inventor's certificate issued before the date of this application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months prior to this application and that as to applications for patent or inventor's certificate filed by me or my legal representatives or assigns in any country foreign to the United States of America, the earliest filed foreign application(s) filed within twelve months prior to the filing date of this application and all foreign applications filed more than twelve months prior to the filing date of this application, if any, are identified below.

CHECK APPROPRIATE BOX:

____   no earlier-filed foreign applications.

___   Required information as to foreign applications filed prior to the filing date of this application is on page ___ attached hereto and made a part hereof.

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

NAME	REGISTRATION NO.
Peter Rutkowski	32,627
J.P. Violette	33,042
Dan A. Shifrin	34,473
James J. Murphy	34,503
Ehrlich, Henry L.	39,663
Garsson, Ross S.	38,150
Kordzik, Kelly K.	36,571
Lando, Peter R.	P 45,513
Mason, Dwayne	38,959
Murphy, James J.	34,503
Naifeh, Bill R.	44,962
Newberger, Barry	41,527
Rogers, Charles J.	38,286
Shaddox, Robert C.	34,011

SEND CORRESPONDENCE TO:
WINSTEAD SECHREST & MINICK
5401 Renaissance Tower
1201 Elm
Dallas, Texas 75270

DIRECT TELEPHONE CALLS TO:
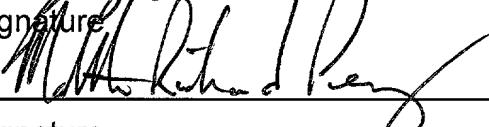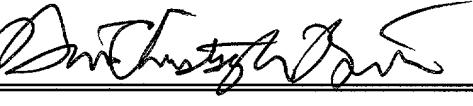
James J. Murphy, Esq.
(214) 745-5374
Fax: (214) 745-5390

Winstead Sechrest & Minick, P.C.'s customer number is 23-2426.

(201) FULL NAME OF INVENTOR	LAST NAME NORTH	FIRST NAME Gregory	MIDDLE NAME Allen	
RESIDENCE & CITIZENSHIP	CITY Austin	STATE OR FOREIGN COUNTRY Texas	COUNTRY OF CITIZENSHIP United States of America	
POST OFFICE ADDRESS	POST OFFICE ADDRESS 804 Weston Lane	CITY Austin	STATE OR COUNTRY Texas, USA	ZIP CODE 78733
(202) FULL NAME OF INVENTOR	LAST NAME PERRY	FIRST NAME Matthew	MIDDLE NAME Richard	

RESIDENCE & CITIZENSHIP	CITY Austin	STATE OR FOREIGN COUNTRY Texas	COUNTRY OF CITIZENSHIP United States of America	
POST OFFICE ADDRESS	POST OFFICE ADDRESS 7913 High Hollow Drive	CITY Austin	STATE OR COUNTRY Texas, USA	ZIP CODE 78750
(203) FULL NAME OF INVENTOR	LAST NAME KIRCHER	FIRST NAME Brian	MIDDLE NAME Christopher	
RESIDENCE & CITIZENSHIP	CITY Round Rock	STATE OR FOREIGN COUNTRY Texas	COUNTRY OF CITIZENSHIP United States of America	
POST OFFICE ADDRESS	POST OFFICE ADDRESS 2204 Frontier Trail	CITY Round Rock	STATE OR COUNTRY Texas, USA	ZIP CODE 78681

I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Name (201) Gregory Allen North	Signature	Date 1/31/00
Name (202) Matthew Richard Perry	Signature	Date 1/31/00
Name (203) Brian Christopher Kircher	Signature	Date 1/31/00

::ODMA\PCDOCS\DALLAS_1\3248512\1

Section 1.56 Duty to Disclose Information Material to Patentability

(a)   A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the office all information known to that individual to be material to patentability as defined in this section. The duty to disclose information exists with respect to each pending claim until the claim is canceled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is canceled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclose all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by Sections 1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applications to carefully examine:

(1)   prior art cited in search reports of a foreign patent office in a counterpart application, and

(2)   the closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b)   Under this section, information is material to patentability when it is not cumulative to information already of record of being made of record in the application, and

(1)   It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim; or

(2)   It refutes, or is inconsistent with, a position the application takes in:

(i)   opposing an argument of unpatentability relied on by the Office, or

(ii)   Asserting an argument of patentability.

A prima facie case of patentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any considerations given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

©   Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1)   Each inventor named in the application;

(2)   Each attorney or agent who prepares or prosecutes the application; and

(3)   Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d)   Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent or inventor.

::ODMA\PCDOCS\DALLAS_1\3248512\1
233:2836-G99994

WSM 2836-P100US   -4-